PB88-202395

Dispersion Model for Elevating Dense Gas Jet Chemical Releases
Volume 2. User's Guide

Arkansas Univ., Fayetteville

Prepared for

Environmental Protection Agency
Research Triangle Park, NC

Apr 88

U.S. Department of Commerce
National Technical Information Service
NTIS

TECHNICAL REPORT DATA

(Please read Instructions on the reverse before completing)

1. REPORT NO. EPA/SW/DK-88/048b
EPA 450/4-88-006b

3. RECIPIENT'S ACCESSION NO.
PB88-202395

4. TITLE AND SUBTITLE
A Dispersion Model for Elevated Dense Gas Jet Chemical Releases--Volume II, User's Guide

5. REPORT DATE
April 1988

6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)
Dr. Jerry Havens

8. PERFORMING ORGANIZATION REPORT NO.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

10. PROGRAM ELEMENT NO.

11. CONTRACT/GRANT NO.
P.O. #6D2746NASA

12. SPONSORING AGENCY NAME AND ADDRESS
U.S. Environmental Protection Agency
Office of Air Quality Planning and Standards
Source Receptor Analysis Branch
Research Triangle Park, N.C. 27711

13. TYPE OF REPORT AND PERIOD COVERED

14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES For Diskettes, see: PB88-202379
EPA Project Officer: Dave Guinnup

16. ABSTRACT

This document is the second of two volumes describing the development and use of a computer program designed to model the dispersion of heavier-than-air gases which are emitted into the atmosphere with significant velocity through elevated ports. Volume II addresses the user aspects of the program, discussing model inputs and outputs and describing the installation of the code onto the user's VAX computer. An example simulation is carried out to provide the user with a benchmark for the model's operation.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group
Air pollution Dense gas Mathematical model Computer model	Dispersion Elevated sources	

18. DISTRIBUTION STATEMENT
Release unlimited

19. SECURITY CLASS *(This Report)*

20. SEC CLASS *(This page)*

21. NO. OF PAGES
377

22. PRICE
A17 32.95

EPA Form 2220-1 (Rev. 4-77) PREVIOUS EDITION IS OBSOLETE

United States
Environmental Protection
Agency

Office of Air Quality
Planning and Standards
Research Triangle Park NC 27711

EPA-450/4-88-006b
April 1988

PB88-202395

Air

EPA

A Dispersion Model For Elevated Dense Gas Jet Chemical Releases

Volume II. User's Guide

REPRODUCED BY
U.S. DEPARTMENT OF COMMERCE
National Technical Information Service
SPRINGFIELD, VA. 22161

EPA-450/4-88-006b
April 1988

A Dispersion Model For Elevated Dense Gas Jet Chemical Releases

Volume II. User's Guide

U.S. ENVIRONMENTAL PROTECTION AGENCY
Office of Air and Radiation
Office of Air Quality Planning and Standards
Research Triangle Park, North Carolina 27711

DISCLAIMER

This report has been reviewed by the Office of Air Quality Planning and Standards, U.S. Environmental Protection Agency, and approved for publication as received from Dr. Jerry Havens. Approval does not signify that the contents necessarily reflect the views and policies of the U. S. Environmental Protection Agency, nor does mention of trade names or commercial products constitute endorsement or recommendation for use. Copies of this report are available from the National Technical Information Service (NTIS).

ACKNOWLEDGEMENTS

The elevated dense gas jet model incorporates methodology published by Ooms and his colleagues at the Technological University Delft, The Netherlands, along with the DEGADIS dense gas dispersion model developed at the University of Arkansas. Tom Spicer, my coauthor of DEGADIS, contributed to the development and was responsible for the modifications to DEGADIS required for interfacing with Ooms' model.

Jerry Havens

PREFACE

This version of the elevated dense gas dispersion model, Ooms/DEGADIS, has been developed by Dr. Jerry Havens and Dr. Tom Spicer of the University of Arkansas with the support of funding from the United States Environmental Protection Agency (EPA). It represents intermediate development of a dense gas modeling package which is undergoing further refinement through additional EPA support. While this model has not been extensively tested against field data, and is subject to specific limitations and uncertainties, the EPA is making it publicly available through the National Technical Information Service (NTIS) as an interim research tool pending further model evaluation and development.

The Ooms/DEGADIS model has been written in FORTRAN with specific intent for compilation and execution on a Digital Equipment Corporation VAX computer. Implementation of this model on any other computer system may be attempted at the risk of the user. Considerations for such implementation, however, are discussed in Appendix B of Volume II.

To facilitate dissemination of the model, it is being provided on two PC-compatible diskettes. The model should be uploaded via modem from a PC terminal to a host VAX computer, and several files must then be renamed prior to compilation and execution. Specific information on this process is contained in the file AAREADME.TXT. Print this file and the compilation batch file, BUILD.COM, prior to attempting compilation.

It is the concern of the EPA that this model be applied only within the framework of its intended use. To this end the user is referred to the specific recommendations in Volume I, Section VII for model application. These recommendations take advantage of the fact that, in this version of the Ooms/DEGADIS model, the portion of the model adapted from Ooms and his colleagues can be executed as a standalone model, as can the DEGADIS portion. To begin any particular simulation, it is recommended that the Ooms portion of the model be executed by itself.

This can be accomplished by setting the input variable <TEND> equal to 1. If the output from this simulation predicts that the plume will touch down less than 1 kilometer from the source, the complete Ooms/DEGADIS model may be appropriately applied (set <TEND> equal to zero or greater). If the plume is not predicted to touch down within 1 kilometer, this model should not be used.

VOLUME II

TABLE OF CONTENTS

Section — Page

List of Tables iii

List of Figures v

List of Symbols vii

Summary ix

I. Model Summaries 1
  - Ooms Model 1
  - DEGADIS Model 3
  - Limitations and Cautions 6

II. Model Inputs 9
  - VAX/VMS Command Procedure 9
  - Simulation Definition 11

III. Model Implementation 19

IV. Example Simulation 21

References 25

Appendix A. Installation on VAX/VMS A-1
B. Considerations for Installation on other Systems B-1
C. DEGADIS Implementation C-1
D. Standalone DEGADIS Example Simulation D-1
E. Ooms' Model Code Listing E-1
F. DEGADIS Code Listing F-1
G. Ooms-DEGADIS Interface Program Code Listing G-1
H. Partial Listing of DEGADIS Variables H-1
I. DEGADIS Diagnostic Messages I-1
J. Ooms/DEGADIS Example Simulation Output J-1

LIST OF FIGURES

Figure		Page
I.1	Schematic Diagram of Ooms' Model	1
I.2	Schematic Diagram of DEGADIS Model	4
II.1	Example DEGADIS Command Procedure on VAX/VMS for a Steady-State Simulation Named TEST_S	10
II.2	Example DEGADIS Command Procedure on VAX/VMS for a Transient Simulation Named TEST	10
II.3	RUN_NAME.IN Structure Required by OOMS_IN	13
C.1	DEGADISIN Flowchart	C-2
C.2	Structure for Free-Formatted RUN_NAME.INP File	C-2
C.3	DEGADIS1 Flowchart	C-3
C.4	SYS$DEGADIS:EXAMPLE.ER1 Listing	C-8
C.5	DEGADIS2 Flowchart	C-11
C.6	SYS$DEGADIS:EXAMPLE.ER2 Listing	C-14
C.7	DEGADIS3 Flowchart	C-16
C.8	SYS$DEGADIS:EXAMPLE.ER3 Listing	C-19
C.9	DEGADIS4 Flowchart	C-21
C.10	Structure of Input for DEGADIS	C-23
C.11	SDEGADIS2 Flowchart	C-25
D.1	BURRO9S.INP Listing	D-14
D.2	BURRO9.INP Listing	D-15

LIST OF TABLES

Table		Page
II.1	Representative Monin-Obukhov Lengths and Power Law Exponents for Different Atmospheric Stabilities	12
D.1	Summary of Burro 9 Test Conditions Used in Example Simulations	D-1

LIST OF SYMBOLS

b_j	characteristic width of plume (radius = $b_j\sqrt{2}$), m
c	local concentration, kg/m^3
c^*	concentration on plume axis, kg/m^3
r	radial distance to jet/plume axis, m
s	distance along plume axis, m
u	local velocity in direction of plume axis, m/s
u_a	wind velocity, m/s
u^*	plume excess velocity at plume axis, $u(r = 0) - u_a\cos\theta$, m/s
x	horizontal coordinate, m
y	vertical coordinate, m
α_1, α_2, α_3	entrainment coefficients
θ	angle between plume axis and horizontal, radius
λ	turbulence Schmidt number, 1.16
ρ	local density, kg/m^3
ρ_a	air density, kg/m^3
ρ^*	plume excess density at plume axis, $\rho(r = 0) - \rho_a$, kg/m^3

SUMMARY

The mathematical modeling techniques used to predict atmospheric dispersion of denser-than-air gases in the Ooms and DEGADIS models are briefly summarized. The Ooms model describes the release and subsequent dilution and trajectory of an elevated gas jet as a gas plume. If the plume falls to ground level, DEGADIS describes the resulting ground-level plume. As well, DEGADIS can be used to describe the release and dilution from a low-momentum, ground-level release. The necessary model-input information to simulate a denser-than-air gas release is summarized. Example simulations of steady-state and transient release are included. Guidelines for installation of the models are included, and a listing of the Ooms and DEGADIS models are included along with a partial list of program variables and diagnostic messages.

I. MODEL SUMMARIES

This section is intended to summarize the critical components of the formulation of each model and the associated limitations and cautions. The suggested limitations and guidelines are based on the experience gained during the development of the models and verification by comparison with a wide range of denser-than-air gas laboratory and field-scale dispersion tests.

Ooms' Model

Ooms, Mahieu, and Zelis' (1974) model comprises simplified balance equations for mass and momentum to describe the jet illustrated in Figure I.1.

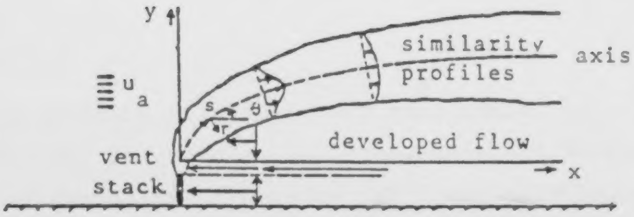


Figure I.1. Schematic diagram of Ooms' model.

Gaussian similarity profiles for velocity, density, and concentration are assumed to apply in the developed jet:

$$u(s,r,\theta) = u_a \cos\theta + u^*(s)\, e^{-r^2/b_j^2(s)} \quad (I.1)$$

$$\rho(s,r,\theta) = \rho_a + \rho^*(s)\, e^{-r^2/\lambda^2 b_j^2(s)} \quad (I.2)$$

$$c(s,r,\theta) = c^*(s)\, e^{-r^2/\lambda^2 b_j^2(s)} \quad (I.3)$$

Balance equations for mass, horizontal and vertical momentum, and energy are integrated over the radius of the plume, and the resulting ordinary differential equations are numerically integrated. Initial conditions are specified at the beginning of the "developed flow" region of the jet (Figure I.1). The trajectory of the jet to the developed flow region is calculated using wind-tunnel data correlations by Kamotani and Greber (1972).

The balance equations for mass and momentum incorporate empirical coefficients for estimating air entrainment. The coefficients α_1, α_2, and α_3 provide for entrainment as follows:

α_1 is the entrainment coefficient for a turbulent free jet. The value 0.057 was incorporated by Ooms, after Albertson, et al. (1950).

α_2 accounts for entrainment into the plume at a sufficiently long distance downwind of the vent where the velocity of the plume approaches the wind velocity. The value 0.5 was incorporated by Ooms, after Richards (1963).

α_3 accounts for entrainment due to atmospheric turbulence. Ooms suggested estimation of the entrainment velocity as $u' = (\epsilon b_j)^{1/3}$, with specification of ϵ (the eddy energy dissipation) as a function of height, wind velocity, and atmospheric stability. The eddy energy dissipation for a neutral atmosphere was recommended by Briggs (1969):

$\epsilon = 0.0677\ u_a/z\ (m^2/s^3)$ for $z < 300$ m

and for "unstable" and "stable" atmospheres by Kaimal et al. (1976):

$\epsilon = 0.004\ (m^2/s^3)$ for unstable atmospheres

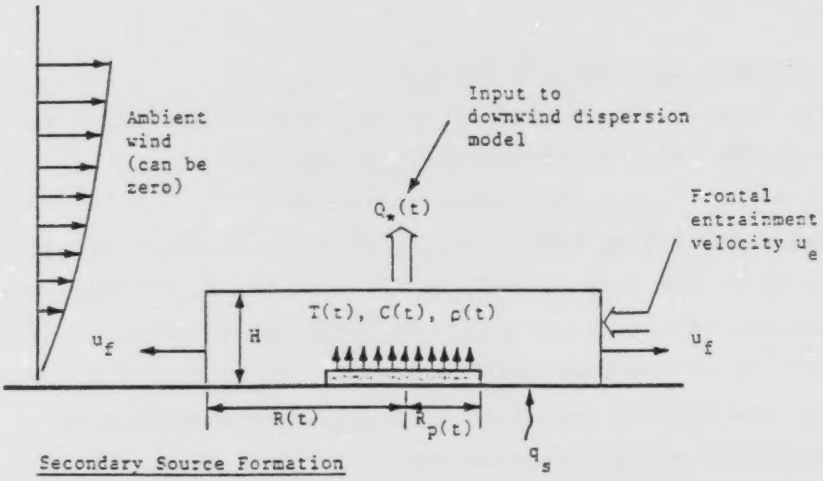
$\epsilon = 0$ for stable atmospheres

The value 1.0 for α_3 was incorporated by Ooms, after Briggs (1969).

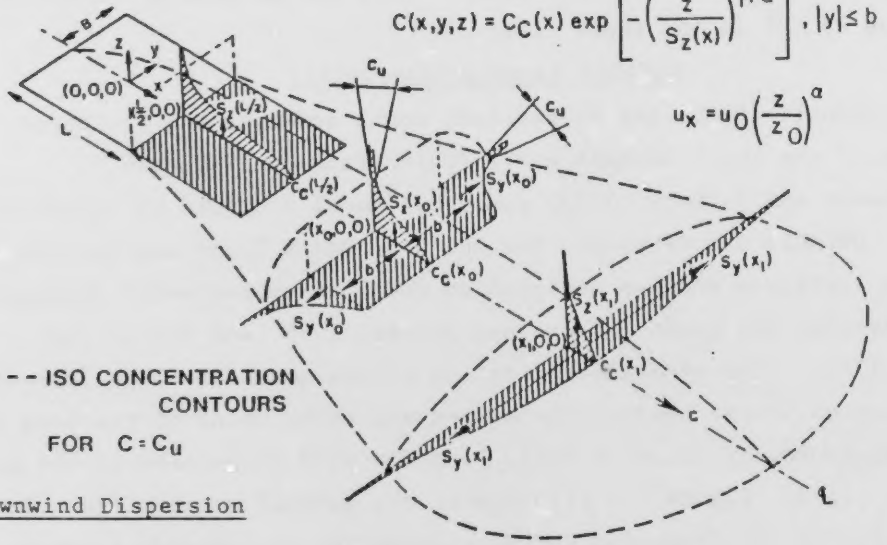
A listing of the code is included in Appendix E. Because the Ooms model equations were derived using the ideal gas equation of state, the current version of the Ooms model code does not account for changes in temperature and density associated with moisture condensation or with contaminant phase changes (contaminant aerosol). Also, the Ooms model will not approach the Gaussian plume model as the density and momentum of the released jet approach ambient values. This is due to the fact that the Ooms model assumes the jet to be radially symmetric at all times which is not true for the Gaussian plume model since σ_y and σ_z are not generally equal. Finally, because the Ooms model code uses fixed-step integration routines, duplicate simulations should be made with different step sizes varying by a factor of about 3 to 5 to ensure the **numerical error is sufficiently small. This can be done by adjusting the value of <H> in the input file.**

DEGADIS Summary Description

The DEGADIS (DEnse GAs DISpersion) model combines the principal features of the Shell HEGADAS model (Colenbrander, 1980, and Colenbrander and Puttock, 1983) and a box model proposed by van Ulden (1983). DEGADIS was developed for the U.S. Coast Guard and the Gas Research Institute and was designed to model the atmospheric dispersion of denser-than-air gases (Havens and Spicer, 1985, and Spicer and Havens, 1987). The general application of the model involves formation of a "secondary" gas source, the subsequent entrainment of gas from that secondary source by the wind field, and downwind dispersion of the gas plume or cloud. Figure I.2 illustrates the general methodology. The description of the formation and development of the secondary source utilizes a box model. The entrainment from the secondary source and subsequent downwind dispersion utilizes the similarity representations of the cloud concentration and vertical velocity profiles of the HEGADAS model. Denser-than-air gas releases which cannot be represented as


Ambient wind (can be zero)
Input to downwind dispersion model
Q*(t)
Frontal entrainment velocity u_e
T(t), C(t), ρ(t)
H
u_f
u_f
R(t)
R_p(t)
q_s

Secondary Source Formation


B
z
y
x
(0,0,0)
(L/2,0,0)
S_z(L/2)
L
C_c(L/2)
c_u
c_u
S_y(x_0)
S_z(x_0)
(x_0,0,0)
b
b
C_c(x_0)
S_y(x_0)
S_y(x_1)
S_z(x_1)
(x_1,0,0)
C_c(x_1)
c
S_y(x_1)
℄
- - - ISO CONCENTRATION CONTOURS
FOR C = C_u

$$C(x,y,z) = C_C(x)\exp\left[-\left(\frac{|y|-b(x)}{S_y(x)}\right)^2-\left(\frac{z}{S_z(x)}\right)^{1+\alpha}\right],\ |y|>b$$

$$C(x,y,z) = C_C(x)\exp\left[-\left(\frac{z}{S_z(x)}\right)^{1+\alpha}\right],\ |y|\leq b$$

$$u_x = u_0\left(\frac{z}{z_0}\right)^{\alpha}$$

Downwind Dispersion

Figure I.2. Schematic diagram of DEGADIS model.

steady, continuous releases are modeled as a series of pseudosteady releases. A complete description of DEGADIS can be found in Havens and Spicer (1985) or Spicer and Havens (1987).

Application of the model to releases of a denser-than-air gas in zero wind involves only the box model. The box model treatment of gravity spreading and associated air entrainment is based on parameterization of the laboratory still-air experiments described by Havens and Spicer (1985). For releases in wind, the box model also describes the source development but provides for entrainment of the gas-air source cloud into the ambient wind field.

DEGADIS incorporates heat transfer and water transfer when applicable from the underlying surface to the cloud. Inclusion of these procedures in the model is optional. Effects of heat transfer on both the mean cloud buoyancy and the vertical turbulent mixing (air entrainment) are included while direct effect of water transfer is included only in the mean cloud buoyancy.

DEGADIS is written in Digital Equipment Corporation's VAX/VMS* Fortran (a superset of ANSI Fortran 77); it is composed of six programs which communicate using ASCII files (see Section III and Appendix C). A listing of the code is included as Appendix F, and a partial list of program variables is given in Appendix H. Considerations for installation of DEGADIS are discussed in Appendices A and B. DEGADIS self-diagnostics are listed in Appendix I along with suggested actions. (Appendix D discusses the input information for a standalone DEGADIS simulation.)

As a standalone model, DEGADIS application should be limited to the description of atmospheric dispersion of denser-than-air gas releases at ground level onto flat, unobstructed terrain or water. Because of the assumption of flat, unobstructed terrain, when the surface roughness used in DEGADIS becomes a significant fraction of the depth of the dispersing layer, this assumption may no longer be satisfied. Application to releases from sources above ground level (e.g. overflow from dikes) would be expected to give conservative predictions of the downwind hazard zones, but this has not been verified.

---

*VAX and VMS are registered trademarks of Digital Equipment Corporation.

Demonstration of DEGADIS has been primarily directed to the prediction of hazard extent defined by gas concentrations in the hydrocarbon flammable limit range (~1 to 20%). Even though the relation between peak gas concentration and time-averaged gas concentration is uncertain, there is some basis for using 2.0 as an estimate of the peak-to-time-averaged-concentration ratio for determining a flammable gas concentration zone. If this assumption is made, the predicted distance to LFL/2 would be the maximum distance at which a flammable gas concentration would be predicted. Based on the simulations of field experiments presented in Havens and Spicer (1985), the average ratio of observed distance to calculated distance for a given time-averaged concentration level (OBS/PRE) ranged from 0.82 to 1.03 for the 2.5% level nine out of ten times (i.e. 90% confidence interval); for the 5% level, the average (OBS/PRE) ranged from 0.73 to 0.96 for a 90% confidence interval. If for a given release scenario the calculated distance to the 2.5% average concentration level was 120 m, the distance to the 2.5% average concentration for nine out of ten realizations of the same release would be expected to range (on the average) between 98 m and 124 m, which would also represent the range of the downwind extent of the flammable gas concentration zone for LNG if the peak-to-average ratio of 2.0 is assumed.

Limitations and Cautions

There are some items which should be considered when using the Ooms and DEGADIS models. As previously stated, because the Ooms model equations were derived using the ideal gas equation of state, the current version of the Ooms model code does not account for changes in temperature and density associated with moisture condensation or with contaminant phase changes (contaminant aerosol). Of course, neither model is currently capable of determining any interactions with obstacles in the flow. Also, the Ooms model will not approach the Gaussian plume model as the density and momentum of the released jet approach ambient values. This is due to the fact that the Ooms model assumes the jet to be radially symmetric at all times which is not true for the Gaussian plume model since σ_y and σ_z are not generally equal. Finally, because the Ooms model code uses fixed-step integration routines, duplicate simulations should be made with different step sizes varying by a factor of about 3 to 5 to ensure the numerical error is sufficiently small.

The Ooms model portion of the calculation is terminated when $(c/c^*) = 0.1$ at ground level, and the output from the Ooms model is used to establish the initial conditions for DEGADIS. In DEGADIS, the source concentration is set to c^*, and the source radius is set to $\sqrt{2}b_j$. Because the Ooms model calculation is stopped when $(c/c^*) = 0.1$ at ground level, the jet x-direction momentum may still be significant. Because DEGADIS describes low initial momentum releases, care should be taken that the jet x-direction momentum is no longer significant. For cases when the jet x-direction momentum is not significant when $(c/c^*) = 0.1$ at ground level, the method of first-order lines has been proposed to account for the ground interaction (Dodge et al., 1982). This has not been implemented since the maximum concentration as a function of distance may increase over the region of ground interaction for this method (which is physically impossible).

Preceding page blank

II. MODEL INPUTS

As implemented under VAX/VMS, the Ooms/DEGADIS models use three areas of input information:

- VAX/VMS command procedure for execution
- simulation definition
- numerical parameters.

The VAX/VMS command procedure used to execute Ooms/DEGADIS is generated by OOMS_IN. As well, OOMS_IN is the input module which reads the simulation definition. An example input session is included in Section IV. The numerical parameters (convergence criteria, initial increments, etc.) are supplied to DEGADIS though a series of input files. Although these numerical parameters are easily changed, the user should need to change these only rarely with the exception of the time-sort parameters which are explained in Section IV.

VAX/VMS Command Procedure

The VAX/VMS command procedure generated by OOMS_IN controls the execution of images for the simulation. Image execution follows one of two paths, either for a transient (time-limited) release or for a steady-state release. OOMS_IN will automatically generate the appropriate command procedure; but first, OOMS_IN requires a simulation name be specified. The simulation name must be a valid VAX/VMS file name without a file extension and is designated RUN_NAME. OOMS and DEGADIS will use this file name with standard extensions for input, interprocess communication, and output. Figures II.1 and II.2 show example VAX/VMS command procedures for the run name TEST_S and TEST for steady-state and transient releases, respectively. The directory which contains the executable images of the Ooms model and DEGADIS has been assigned the system logical name SYS$DEGADIS (see Appendix A). The COPY/LOG command simply copies a file from the first argument to the second argument, and the RUN command executes the specified image. Of course, these steps may also be carried out by issuing the commands at a terminal.

```
$ ASSIGN TEST_S.INO FOR001
$ ASSIGN TEST_S.OUT FOR003
$ ASSIGN TEST_S.IND FOR002
$ RUN SYS$DEGADIS:OOMS
$ DEASSIGN FOR001
$ DEASSIGN FOR002
$ DEASSIGN FOR003
$ RUN SYS$DEGADIS:DEGBRIDGE
$ TEST_S
$ COPY/LOG   SYS$DEGADIS:EXAMPLE.ER1   TEST_S.ER1
$ COPY/LOG   SYS$DEGADIS:EXAMPLE.ER2   TEST_S.ER2
$ RUN   SYS$DEGADIS:DEGADIS1
TEST_S
$ RUN   SYS$DEGADIS:SDEGADIS2
TEST_S
$ COPY/LOG   TEST_S.OUT+TEST_S.SCL+TEST_S.SR3 -
TEST_S.LIS
```

Figure II.1. Example DEGADIS command procedure on VAX/VMS for a steady-state simulation named TEST_S.

```
$ ASSIGN TEST.INO FOR001
$ ASSIGN TEST.OUT FOR003
$ ASSIGN TEST.IND FOR002
$ RUN SYS$DEGADIS:OOMS
$ DEASSIGN FOR001
$ DEASSIGN FOR002
$ DEASSIGN FOR003
$ RUN SYS$SYSDEGADIS:DEGBRIDGE
TEST
$ COPY/LOG   SYS$DEGADIS:EXAMPLE.ER1   TEST.ER1
$ COPY/LOG   SYS$DEGADIS:EXAMPLE.ER2   TEST.ER2
$ COPY/LOG   SYS$DEGADIS:EXAMPLE.ER3   TEST.ER3
$ RUN   SYS$DEGADIS:DEGADIS1
TEST
$ RUN   SYS$DEGADIS:DEGADIS2
TEST
$ RUN   SYS$DEGADIS:DEGADIS3
TEST
$ COPY/LOG   TEST.OUT+TEST.SCL+TEST.SR3 -
TEST.LIS
```

Figure II.2. Example DEGADIS command procedure on VAX/VMS for a transient simulation named TEST.

Simulation Definition

OOMS_IN is a method of simulation definition where the user specifies information about the ambient wind field, the properties of the released gas, and some details of the release.

The input is carried out using free-formatted files. The use of this method means that a value of each parameter must be specified in the input file even if it has no meaning for the simulation at hand. As well, this form of input allows the user creating the input file to be able to ignore spacing in the input file. Numbers on a particular line can be separated by commas, spaces, or tabs. Comments may also be included at the end of any line in the input file as long as all required values are given before any comments are included. Lines which are only comments are not allowed. (Sample input files are included in Section IV.)

The ambient wind field is characterized by a known velocity u_0 at a given height z_0, a surface roughness z_R, and the Pasquill stability class or Monin-Obukhov length. The Pasquill stability class is used to estimate values of the lateral similarity parameter coefficients δ and β (along with the averaging time as discussed in Spicer and Havens, 1987) and values of the along-wind similarity coefficients (Beals, 1971). The Monin-Obukhov length λ used by Businger et al. (1971) in their logarithmic velocity profile function ψ (Table II.1) is used to calculate the friction velocity u_*. In addition to these specifications, the ambient temperature, pressure, and humidity must be specified.

The properties of air and the released gas are used to evaluate the mixture density as a function of temperature and composition. The desired released gas properties include the molecular weight MW_c, the release temperature T_0, and two constants q_1 and p_1 which describe the heat capacity according to the equation

$$C_{p_c}(T) = (MW_c)^{-1} \left[3.33 \times 10^4 + q_1 \left[\frac{T^{p_1} - T_0^{p_1}}{T - T_0} \right] \right] \qquad \text{(II.1)}$$

TABLE II.1
REPRESENTATIVE MONIN-OBUKHOV LENGTHS AND POWER LAW EXPONENTS FOR DIFFERENT ATMOSPHERIC STABILITIES

Pasquill Stability Category	Monin-Obukhov Length (λ) as a Function of Surface Roughness z_R(m)	Typical (*) Power Law Exponents α	Corrections to Logarithmic Profiles as Given by Businger (1973)
A	$-11.4\ z_R^{0.10}$	0.108	$\psi = 2 \ln \left[\frac{1+a}{2}\right] + \ln \left[\frac{1+a^2}{2}\right]$ $- 2 \tan^{-1}(a) + \pi/2$ with $a = (1 - 15(Z/\lambda))^{1/4}$
B	$-26.0\ z_R^{0.17}$	0.112	
C	$-123\ z_R^{0.30}$	0.120	
D	∞	0.142	$\psi = 0$
E	$123\ z_R^{0.30}$	0.203	$\psi = -4.7(Z/\lambda)$
F	$26.0\ z_R^{0.17}$	0.253	

(*) calculated for 8 m/s at 10 m with surface roughness of 0.001 m where $C_{p_c}(T)$ is the mean heat capacity (J/kg K) at temperature T. A constant heat capacity can also be specified as discussed later in this section.

In specifying the details of the release, the user must choose to simulate the release as time-limited (transient) or steady-state. As well, the jet elevation, diameter, and orientation must be specified.

Figure II.3 summarizes the structure of the input file RUN_NAME.IN needed by OOMS_IN. The following description is written for each of the variables in Figure II.3. Variables are written in brackets. After each line of input, an explanation is included. All units are SI (meters, kilograms, second) except as specified.

```
<TITLE1>
<TITLE2>
<TITLE3>
<TITLE4>
<UO>   <ZO>
<ZR>
<INDVEL> <ISTAB> <MOLEN>
<TAMB> <PAMB> <RELHUM>
<TSURF>
<GASNAM>
<GASMW>
<AVTIM>
<JETTEM>
<GASUL> <GASLL> <ZLL>
<INDHT> <CPC> <CPP>
<ERATE>
<JETELE> <JETDIA>
<TEND>
<XO> <H> <DISTA> <DISTAN>
<K1> <K2> <K3>
```

Figure II.3. RUN_NAME.IN structure required by OOMS_IN.

<TITLE1>

<TITLE2>

<TITLE3>

<TITLE4> Each of <TITLE1>, <TITLE2>, <TITLE3>, and <TITLE4> are used to keep a title block with the output of the model(s). Each are up to 80 characters long.

<U0> <Z0>

<U0> is the ambient wind velocity at <Z0>.

<ZR> <ZR> is the surface roughness.

<INDVEL> <ISTAB> <MOLEN>

This line is used to specify the ambient velocity profile. <INDVEL> is the indicator used to specify whether the ambient velocity profile is based on:

(1) the Pasquill stability category in <ISTAB> using 1 for A, 2 for B, etc.; or

(2) the Monin-Obukhov length in <MOLEN> (as a real number). If a Monin-Obukhov length of infinity is desired, input a value of 0.0 for <MOLEN>.

Note that the Pasquill stability category must be specified (regardless of <INDVEL>) since <ISTAB> is used to estimate other parameters.

<TAMB> <PAMB> <RELHUM>

This line specifies the ambient air temperature (<TAMB> in K), the ambient pressure (<PAMB> in N/m^2 or atm), and the ambient relative humidity (<RELHUM> as a percent).

<TSURF>

This line specifies the surface temperature (in K). If the value of <TSURF> is below 250 K, <TSURF> is set to <TAMB>.

<GASNAM>

<GASNAM> is a user-generated 3-letter name used to identify the dispersing gas.

<GASMW>

<GASMW> is the gas molecular weight (kg/kmole).

<AVTIM>

<AVTIM> is the averaging time to be used for this particular gas. At present, this parameter is used to estimate the value of DELTA in DEGADIS.

<JETTEM>

<JETTEM> is the temperature of the released gas jet (K).

<GASUL> <GASLL> <ZLL>

<GASUL> and <GASLL> are the upper and lower limits (in mole fraction) used for contour computations at the elevation <ZLL>. Note that the computations will be carried out to <GASLL>/2.

<INDHT> <CPC> <CPP>

<INDHT> is used to determine whether heat transfer with the ground is included in DEGADIS. Heat transfer is not included when <INDHT> is set to zero but is included when <INDHT> is nonzero. <CPC> and <CPP> are used to calculate the gas heat capacity using the specification in DEGADIS as a function of temperature. (If this function is used, the average gas temperature is used to specify the mean (constant) heat capacity in the Ooms model.) If a constant contaminant heat capacity is desired, <CPP> is set to 1.0 and <CPC> is set to the desired heat capacity (in J/kg K).

<ERATE>

<ERATE> is the mass evolution rate of the pure contaminant (in kg/s).

<JETELE> <JETDIA>

<JETELE> is the initial jet elevation above ground level.
<JETDIA> is the initial diameter of the jet.

<TEND> <TEND> is the duration of the release. For a steady-state case, input 0.0 for <TEND>. If the Ooms model is to be run alone, input a negative number for <TEND>.

<X0> <H> <DISTA> <DISTAN>

<X0> is the starting value for the jet trajectory integration. <H> is the step size for the Ooms model.
<DISTA> is the downwind distance to the first output value.
<DISTAN> is the distance between output points in the Ooms model.

<K1> <K2> <K3>

<K1>, <K2>, and <K3> are the parameters used to specify the initial jet direction as follows:

<K1> is used for horizontal jets directed upwind (-1) and downwind (1); <K1>=0 for other orientations.

<K2> is used for vertical jets directed upward (+1) and downward (-1); <K2>=0 for other orientations.

<K3> is used for horizontal jets directed transverse to the left (-1) and to the right (+1); <K3>=0 for other orientations.

Note that only one of <K1>, <K2>, or <K3> can be nonzero at any time.

Appendix G contains program listings for OOMS_IN and DEGBRIDGE. (DEGBRIDGE takes the output from the Ooms model and creates the file necessary for DEGADIS to complete the calculations as needed.)

Preceding page blank

III. MODEL IMPLEMENTATION

The models described in Section I have been implemented in VAX/VMS Fortran (a superset of Fortran 77) in the codes OOMS and DEGADIS. DEGADIS is comprised of six separate programs as follows:

(*) DEGADISIN	is the interactive input module which defines the simulation.
(*) DEGADIS1	determines α and describes the gas source for transient and steady-state releases.
(*) DEGADIS2	describes the pseudosteady-state downwind dispersion of the released gas.
(*) DEGADIS3	sorts the results of DEGADIS2 for a transient release at given times.
(*) DEGADIS4	sorts the results of DEGADIS2 for a transient release at given positions.
(*) SDEGADIS2	describes the steady-state downwind dispersion of the released gas.

As indicated in Figures II.1 and II.2, a steady-state release is simulated by executing OOMS, DEGADISIN, DEGADIS1, and SDEGADIS2, while a time-limited (transient) release is simulated by executing OOMS, DEGADISIN, DEGADIS1, DEGADIS2, and DEGADIS3.

Preceding page blank

IV. EXAMPLE SIMULATION

The example simulation conditions shown in Table IV.1 for the accidental release of methylisocyanate (MIC) on December 3, 1984, in Bhopal, India, were reported by Singh (1986). To simulate this release, the MIC was assumed to be released as a pure, ambient temperature gas at a steady rate of 6.72 kg/s. For the lowest concentration of interest, the lethal concentrations to 50% of laboratory animals exposed to MIC (LC_{50}) for one and two hours are about 30 and 20 ppm, respectively.

TABLE IV.1

EXAMPLE SIMULATION RELEASE CONDITIONS

Mass of MIC released	40 tons
Duration of release	90 min
Vent elevation	33 m
Vent diameter	0.2 m
Wind velocity at 10 m	2.9 m/s
Atmospheric stability	E/F

The input file used to simulate the release conditions of Table IV.1 as a steady-state release is shown in Figure IV.1. Note that comments can be included at the end of each line as long as the specified values are entered on the line first; as well, comments can appear at the end of the file after all values have been entered. A surface roughness of 0.1 m has been used. (Note that the surface roughness should not be larger than the depth of the denser-than-air gas layer in DEGADIS; for this case, the value of S_z at the beginning of the steady-state DEGADIS calculation was 33.8 m.) Also, an averaging time of 3600 s has been used to correspond with the averaging time for the LC_{50}; note that the concentrations of 30 ppm (GASUL0.00003 mole fraction) and 20 ppm (GASLL=0.00002 mole fraction) are to be used to calculate concentration contours in DEGADIS. The fact that this is a steady-state simulation is indicated by the value of TEND (0.0). Finally, the jet is assumed to be oriented vertically upward since K1, K2, and K3 are all set to 0.

Note that the Ooms/DEGADIS model can be executed interactively or in batch mode. An example batch command file to run the model under VMS is shown in Figure IV.2 for the example simulation called EXAMPLE. (The logical symbol SYS$DEGADIS: is assigned to the directory which contains the executable image of the model.) Note that any file name may be used; the same file name will be used throughout the simulation with different extensions for internal files used by the model. Upon completion of the simulation, the output of the model is in the file with the original name and extension LIS; for this example, the output file will be EXAMPLE.LIS. To run the model interactively, the same lines shown in Figure IV.2 should be entered at the terminal.

Several lines of output are written to the terminal during the execution of the model. (In batch mode, the same lines are written to the batch log file.) As discussed in Section II.1, the program OOMS_IN reads the input file (EXAMPLE.IN in this case) and generates the input file to the OOMS program; when OOMS_IN finishes, OOMS_IN writes the line "OOMS_IN - beginning command file". At this point, the OOMS program begins; several lines of output are generated showing the values of various parameters calculated by the program. When OOMS finishes, DEGBRIDGE begins; DEGBRIDGE writes the line "DEGBRIDGE - beginning DEGBRIDGE". DEGBRIDGE takes the output of OOMS and generates the necessary input file for DEGADIS. When DEGBRIDGE finishes, the standard DEGADIS procedure begins with the copying of the ER1 and ER2 numerical parameter files. DEGADIS generates several lines of output showing the values of various numerical parameters calculated by the model. The last step of the model is to generate the LIS file from the output files of OOMS and DEGADIS.

For the example case above, the OOMS calculations are terminated at 540 m when the lower edge of the plume reaches ground level. At this point, the centerline concentration (c^*) is 2.83×10^{-3} kg/m^3. From this point, the DEGADIS calculation begins; the initial concentration used in DEGADIS is taken to be 2.83×10^{-3} kg/m^3 (from OOMS) and the value of S_z at the downwind edge of the DEGADIS source is 33.8 m. The output of DEGADIS for this example case shows the 30 ppm level reaches 5.76 km and the 20 ppm level reaches 7.22 km. Note that the downwind distance used in DEGADIS is measured from the DEGADIS source. Therefore, the distances predicted by the OOMS/DEGADIS model to the 30 and 20 ppm concentration levels for the example case are 6.30 km and 7.76 km, respectively. The complete output listing for this example simulation can be found in Appendix J. Note that the "width at z=" to some appropriate mole percent value actually refers to the distance to that isopleth concentration from the plume centerline (i.e., half-width).

The OOMS code can be run (without DEGADIS) using OOMSIN by specifying a TEND which is less than zero. For this case, the input file is identical to the example case shown in Figure IV.1 except that TEND is set equal to -1. As before, the OOMS calculations are terminated at 540 m when the boundary of the cloud first reaches ground level.

Simulation of the release of an impure dense gas can be performed by adjusting the molecular weight and heat capacity inputs to the program. The molecular weight should be the equivalent molecular weight of the mixture at the release point, and the heat capacity inputs should be those of the initial mixture. Concentration (mole fraction) values output by the program should then be multiplied by the initial mole fraction of contaminant in the released mixture to obtain actual mole fraction values.

```
This is a steady-state test simulation of the OOMS and DEGADIS models.
    Methylisocyanate (MIC) release


2.9  10.                                       UO, ZO
0.1                                            ZR
1  6   0.                                      INDVEL, ISTAB, MOLEN
298.  1.  50.                                  TAMB, PAMB, RELHUM
298.                                           TSURF
MIC                                            GASNAM
57.                                            GASMW
3600.                                          AVTIM - based on 1 hr toxic levels
298.                                           JETTEM
0.00003   0.00002  0.5                         GASUL, GASLL, ZLL
0  2000.  1.                                   INDHT,  CPC, CPP
6.72                                           ERATE
33. 0.2                                        JETELE, JETDIA
0.0                                            TEND
1.  1.  10. 10.                                XO, H, DISTA, DISTAN
0  0  0                                        K1, K2, K3

Since ONLY one of {K1, K2, or K3} MUST be nonzero, OOMS_IN checks these values
to ensure that these conditions are met.  If they are not met, OOMS_IN forces
one of {K1, K2, or K3} to be 1; if all are zero, the jet is assumed to be
oriented vertically upward.

This is the end of the file.  Any comments can be included here since the
file is not read after the line for K1 et al. above has been read.
```

Figure IV.1. Listing of EXAMPLE.IN.

```
$RUN SYS$DEGADIS:OOMS_IN
EXAMPLE
```

Figure IV.2. Example command file used to simulate the EXAMPLE simulation.

REFERENCES

Albertson, M. L., Y. B. Dai, R. A. Jenson, and H. Rouse, "Diffusion of Submerged Jets," Transactions of American Society of Civil Engineers, 115, (1950).

Beals, G. A., "A Guide to Local Dispersion of Air Pollutants," Air Weather Service Technical Report 214, April 1971.

Briggs, G. A., "Plume Rise," AEC Critical Review Series," USAEC Division of Technical Information Extension, Oak Ridge, Tennessee, 1969.

Businger, J. A., J. C. Wyngaard, Y. Izumi, and E. F. Bradley, "Flux-Profile Relationships in the Atmospheric Surface Layer," Journal of the Atmospheric Sciences, 28, March 1971.

Colenbrander, G. W., "A Mathematical Model for the Transient Behavior of Dense Vapor Clouds," 3rd International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Basel, Switzerland. 1980.

Colenbrander, G. W. and J. S. Puttock, "Dense Gas Dispersion Behavior: Experimental Observations and Model Developments," International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Harrogate, England, September 1983.

Havens, J. A. and T. O. Spicer, "Development of an Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures," Final Report to U.S. Coast Guard, CG-D-23-80, USCG HQ, Washington, DC, May 1985.

Kaimal, J. C., J. C. Wyngaard, D. A. Haugen, O. R. Cote, and Y. Izumi, "Turbulence Structure in the Convective Boundary Layer," J. Atmos. Sci., 33, 1976.

Kamotani, Yasuhiro and Isaac Greber, "Experiments on a Turbulent Jet in a Cross Flow," AIAA Journal, Vol. 10, No. 11, November 1972.

Ooms, G., A. P. Mahieu, and F. Zelis, "The Plume Path of Vent Gases Heavier than Air," First International Symposium on Loss Prevention and Safety Promotion in the Process Industries, (C. H. Buschman, Editor), Elsevier Press, 1974.

Richards, J. M., "Experiments on the Motion of an Isolated Cylindrical Thermal through Unstratified Surroundings," International Journal of Air and Water Pollution, 7, 1963.

Singh, N. P., presented at G. I. Taylor Centennial Symposium, Cambridge University, England, March 1986.

Spicer, T. O. and J. A. Havens, "Development of Vapor Dispersion Models for Nonneutrally Buoyant Gas Mixtures--Analysis of TFI/NH_3 Test Data," USAF Engineering and Services Laboratory, Draft Final Report, October 1987.

van Ulden, A. P., "A New Bulk Model for Dense Gas Dispersion: Two-Dimensional Spread in Still Air," I.U.T.A.M. Symposium on Atmospheric Dispersion of Heavy Gases and Small Particles, Delft University of Technology, The Netherlands, August 29-September 2, 1983.

APPENDIX A
INSTALLATION ON VAX/VMS

DEGADIS was developed under VAX/VMS V3.5 and VAX-11 Fortran V3.5 although there should be no installation difficulty for VAX/VMS V4.0 or later.

The directory which contains the Fortran source code for DEGADIS must be equivalenced with the logical name SYS$DEGADIS:. If the full directory specification is DQA0:[HACS.DEGADIS], issue the VAX/VMS command:

```
$ ASSIGN DQA0:[HACS.DEGADIS] SYS$DEGADIS:
```

with either the /PROCESS, /GROUP, or /SYSTEM qualifier (/SYSTEM is recommended). Once this assignment is made, the files must be compiled and linked to form DEGADISIN, DEGADIS1, DEGADIS2, DEGADIS3, DEGADIS4, and SDEGADIS2 according to the specifications in Appendix C. The process which compiles and links DEGADIS must have READ, WRITE, and EXECUTE access privileges to SYS$DEGADIS while only READ and EXECUTE access privileges are needed to execute the existing models.

The process for compilation and linking on the VAX is provided in the batch file BUILD.COM. To execute, SUBMIT this file to the batch queue on the VAX computer. The resulting files will be executable according to the instructions in section II of this User's Guide.

APPENDIX B
CONSIDERATIONS FOR INSTALLATION OTHER THAN VAX/VMS

There are two types of problems which may occur when attempting to install DEGADIS on a different computer or operating system. The first source of difficulty is the use of non-standard ANSI Fortran 77 language elements. The second source of difficulty is the use of external VAX/VMS routines in DEGADIS.

The following list is a collection of the VAX-11 Fortran extensions which have been used in DEGADIS:

(*) In-line comments--An exclamation mark (!) is used to include comments at the end of a valid statement.

(*) Special characters--The underscore (_) is used in variable names.

(*) DO loops--DO loops are used with the structure:

```
     DO v = e1,e2[,e3]
      .
      .
      .
     END DO
```

where v is a variable name and e1, e2, and e3 are numeric expressions. The numeric expressions have the standard Fortran 77 meaning.

(*) INCLUDE statements--INCLUDE statements simply allow other source files to be inserted in the routine being compiled at this point in the source. The system table '($SSDEF)' is used to check the status of returning system routines.

(*) OPEN keyword NAME--The OPEN keyword NAME specifies the file name to be opened.

(*) Fortran descriptors--The Q descriptor obtains the integer number of characters remaining in the input record during a READ operation. The dollar sign ($) descriptor suppresses the carriage return at the end of a line on output.

(*) Continuation lines--Continuation lines have been expressed by using either a nonblank character in column 6 or by beginning the line with a tab and a number in the next column.

(*) Concatenation of character strings--Character strings are concatenated using two slashes (//).

The following VAX/VMS subroutines have been used in DEGADIS:

(*) SECNDS

TIME = SECNDS(TIME0)

SECNDS returns to TIME the difference between the number of seconds after midnight on the system clock and the value of TIME0.

(*) LIB$DATE_TIME

ISTAT = LIB$DATE_TIME (STRING)

LIB$DATE_TIME returns a 24-character ASCII string with the system date and time. ISTAT is an integer variable which accepts the return status.

(*) LIB$DO_COMMAND

ISTAT = LIB$DO_COMMAND (STRING)

LIB$DO_COMMAND issues the command STRING (a character string) to VAX/VMS. If the command is not issued, ISTAT contains the failure code. If the command is issued, the calling process is terminated.

APPENDIX C
DEGADIS IMPLEMENTATION

DEGADIS has been implemented in VAX/VMS Fortran (a superset of Fortran 77). DEGADIS is comprised of six separate programs as follows:

(*) DEGADISIN	is the interactive input module which defines the simulation.
(*) DEGADIS1	determines α and describes the gas source for transient and steady-state releases.
(*) DEGADIS2	describes the pseudosteady-state downwind dispersion of the released gas.
(*) DEGADIS3	sorts the results of DEGADIS2 for a transient release for specified times.
(*) DEGADIS4	sorts the results of DEGADIS2 for a transient release for specified positions.
(*) SDEGADIS2	describes the steady-state downwind dispersion of the released gas.

As indicated in Figures II.1 and II.2, a steady-state release is simulated by executing DEGADISIN, DEGADIS1, and SDEGADIS2, while a transient release is simulated by executing DEGADISIN, DEGADIS1, DEGADIS2, and DEGADIS3 (and DEGADIS4 as desired).

Input Module--DEGADISIN

DEGADISIN is the interactive input module which defines the simulation; DEGADISIN is composed of two subroutines (Figure C.1):

(*) DEGADISIN contains the program overhead and generates the command file RUN_NAME.COM which can be used to control simulation execution (F-38).

(*) IOT contains the interactive question-and-answer sequence which defines the simulation; IOT also creates the file RUN_NAME.INP (F-75).

An example of a DEGADISIN query sequence is included in Appendix D. As this information is gathered, it is written to the file RUN_NAME.INP. Once DEGADISIN is completed, RUN_NAME.INP may be edited to correct minor input mistakes. If major revisions are necessary, the recommended practice is to execute DEGADISIN again.

Once the information required by DEGADISIN has been entered properly, DEGADIS may be executed using the command procedure generated by DEGADISIN under the file name RUN_NAME.COM. If DEGADIS is not to be run using this command file, the user must enter the simulation name (RUN_NAME) after each of the programs are begun. As well, the user must provide copies of the numerical parameter files.

DEGADISIN
F-38
IOT
F-75

Figure C.1. DEGADISIN flowchart.

```
                     TITLE(1)
                     TITLE(2)
                     TITLE(3)
                     TITLE(4)
                     U0, Z0, ZR
                     ISTAB
                     DELTA, BETA, ML
                     SIGX_COEFF, SIGX_POW, SIGX_MIN_DIST
                     TAMB, PAMB, HUMID
                     ISOFL, TSURF
                     IHTFL, HTCO
                     IWTFL, WTCO
                     GAS_NAME
                     GAS_MW, GAS_TEMP, GAS_RHOE
                     GAS_CPK, GAS_CPP
                     GAS_UFL, GAS_LFL, GAS_ZSP
                     NP
If (ISOFL≠0) then    DEN((J,1),J=1,5)
(for                 DEN((J,2),J=1,5)
external density        .
calculations)           .
                        .
                     DEN((J,NP),J=1,5)
                     CCLOW
                     GMASSO
                     NT
                     PTIME(1),ET(1),R1T(1),PWC(1),PTEMP(1),PFRACV(1)
                     PTIME(2),ET(2),R1T(2),PWC(2),PTEMP(2),PFRACV(2)
                        .
NT of these             .
                        .
                     PTIME(NT),ET(NT),R1T(NT),PWC(NT),PTEMP(NT),PFRACV(NT)
                     CHECK1, CHECK2, AGAIN, CHECK3, CHECK4, CHECK5
                     TINP
for steady-state     ESS, SLEN, SWID
  only
```

Figure C.2. Structure for free-formatted RUN_NAME.INP file.

Source Module--DEGADIS1

DEGADIS1 estimates values for the friction velocity and ambient wind profile power α and characterizes the primary gas source for the remainder of the model; DEGADIS1 is composed of the following subroutines (Figure C.3):

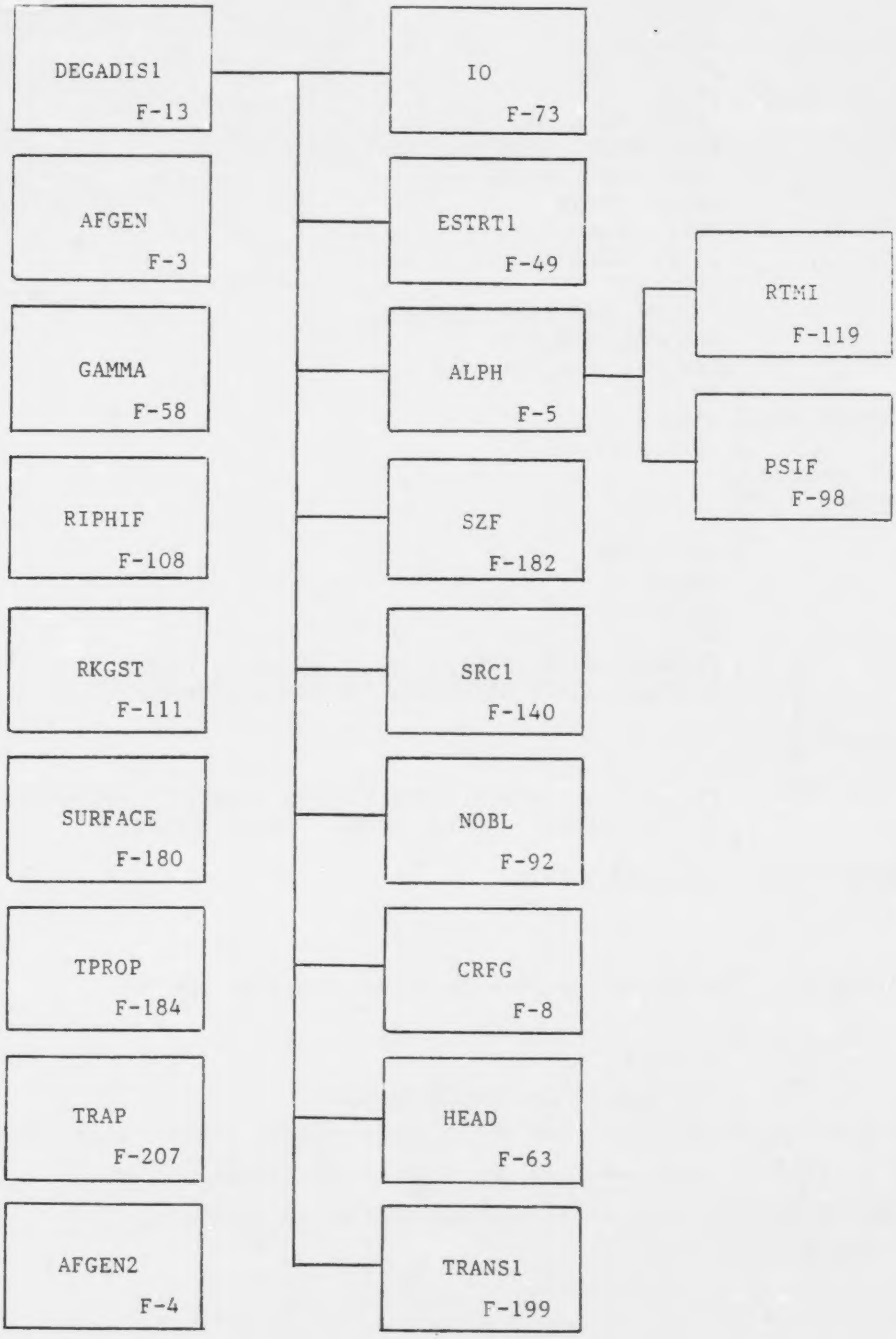

DEGADIS1
F-13
AFGEN
F-3
GAMMA
F-58
RIPHIF
F-108
RKGST
F-111
SURFACE
F-180
TPROP
F-184
TRAP
F-207
AFGEN2
F-4
IO
F-73
ESTRT1
F-49
ALPH
F-5
SZF
F-182
SRC1
F-140
NOBL
F-92
CRFG
F-8
HEAD
F-63
TRANS1
F-199
RTMI
F-119
PSIF
F-98

Figure C.3. DEGADIS1 flowchart.

(*) AFGEN	is a utility which linearly interpolates between a pair of points based on a list of supplied values (F-3).
(*) AFGEN2	is a utility which linearly interpolates between a pair of points based on a list of supplied values (F-4).
(*) ALPH	estimates the ambient wind profile power α by minimizing the integral of the difference between an ambient logarithmic velocity profile and the assumed power law velocity profile (F-5).
(*) CRFG	creates a table of calculated values which will describe the secondary gas source for the downwind dispersion calculations (F-8).
(*) DEGADIS1	contains the program overhead and sequentially calls the routines required to estimate the ambient wind profile power α and to characterize the primary gas source (F-14).
(*) ESTRT1	recovers the numerical parameters contained in the file RUN_NAME.ER1 (F-49).
(*) GAMMA	is a utility function that calculates the gamma function of the argument (i.e. $\Gamma(x)$) (F-58).
(*) HEAD	writes a formatted output heading to the file RUN_NAME.SCL (F-63).
(*) IO	recovers the simulation definition contained in RUN_NAME.INP (F-73).

(*) NOBL	estimates gas source behavior when no gas blanket is present (F-92).
(*) PSIF	calculates the ψ function in the logarithmic velocity profile (F-98).
(*) RIPHIF	is a series of utilities which calculates the Richardson number and the value of $\phi(Ri)$ (F-108).
(*) RKGST	is a utility routine which performs numerical integration of a specified system of equations using a variable-step, modified fourth-order Runge-Kutta method (F-111).
(*) RTMI	is a utility routine which solves the roots of an equation by the Milne method set up by ALPH (F-119).
(*) SRC1	contains the ordinary differential equations which describe the gas blanket formed as a result of the primary gas source (F-140).
(*) SURFACE	is a utility routine which estimates heat and water transfer rates across the bottom surface of the gas layer (F-180).
(*) SZF	estimates the value of S_z if the primary source can just form a gas blanket over the source (F-182).
(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (F-184).
(*) TRANS1	writes the information to continue the next simulation step to the file RUN_NAME.TR2 (F-199).

(*) TRAP	is a utility included for program diagnostics (F-207).

As input, DEGADIS1 requires two files:

(*) RUN_NAME.ER1	contains various numerical parameters. For most simulations, a copy of the SYS$DEGADIS:EXAMPLE.ER1 file will be adequate. (See Figure II.1 or II.2.) A copy of SYS$DEGADIS:EXAMPLE.ER1 is included in Figure C.4.
(*) RUN_NAME.INP	contains the simulation definition as discussed in Appendix D. The format of RUN_NAME.INP is shown in Figure C.2.

As output, DEGADIS1 generates the following files:

(*) RUN_NAME.SCD	contains the calculated values which describe the secondary gas source. It is generated by SRC1 and NOBL and is then read by CRFG; it is a temporary file.
(*) RUN_NAME.SCL	is the listed output which describes the input information for the simulation and the calculated secondary gas source. It is written by HEAD and CRFG.
(*) RUN_NAME.TR2	contains the information to continue the next simulation step.

```
!This is an example of how to set up and use the run parameter
! input files.  Comment lines start with an exclamation mark(!)
! in the first column.  The only restrictions for data input are
! as follows:
!           1) The data must be entered in the same order
!            all of the time.
!           2) Only the number must be between columns 10 and 20.
!           3) Always include the decimal point in the number
!
! Column layout:
!23456789012345678901234567890
!--------1---------2---------3
!        I         I
STPIN     0.01       MAIN - RKGST - INITIAL STEP SIZE
ERBND     0.0025     MAIN - RKGST - ERROR BOUND
STPMX     5.12       MAIN - RKGST - MAXIMUM STEP SIZE
WTRG       1.        MAIN - RKGST - WEIGHT FOR RG
WTTM       1.        MAIN - RKGST - WEIGHT FOR Total Mass
WTYA       1.        MAIN - RKGST - WEIGHT FOR ya
WTYC       1.        MAIN - RKGST - WEIGHT FOR yc
WTEB       1.        MAIN - RKGST - WEIGHT FOR Energy Balance
WTmB       1.        MAIN - RKGST - WEIGHT FOR Momentum Balance
WTuh       1.        MAIN - RKGST - WEIGHT FOR Ueff*Heff
XLI       0.05       ALPH - LOWER LIMIT OF SEARCH FOR ALPHA
XRI       0.40       ALPH - UPPER LIMIT OF SEARCH FOR ALPHA
EPS       0.001      ALPH - ERROR BOUND USED BY 'RTMI'
ZLOW      0.01       ALPHI - maximum BOTTOM HEIGHT FOR FIT OF ALPHA
!
STPINZ    -0.02      ALPHI - INITIAL RKGST STEP <0.
!
ERBNDZ    0.005      ALPHI - ERROR BOUND FOR RKGST
!
STPMXZ    -0.04      ALPHI - MAXIMUM STEP FOR RKGST <0.
!
!  Note that comment lines can be mixed with the numbers.
!
SRCOER    0.007      SRC1O - OUTPUT Error criterion
SRCSS      5.2       SRC1O - min time for Steady; STPMX
SRCcut    .00001     SRC1O - min height for blanket
htcut     .0         SRC1 - min height for blanket heat transfer
ERNOBL    1.0005     NOBL - CONVERGENCE ratio
NOBLPT    100.       NOBL - NUMBER OF POINTS
!                       USED ON THE LAST PORTION OF THE SOURCE
!
crfger    0.008      error criterion in building GEN3 vectors
!
epsilon   0.59       epsilon USED IN AIR ENTRAINMENT SPECIFICATION
!
! /SPRD_CON/
!
ce        1.15       constant in gravity slumping equation
```

Figure C.4. SYS$DEGADIS:EXAMPLE.ER1 listing.

```
delrhomin  0.025     stop cloud spread if delrho<delrhomin
!
!
! /SZFC/
!
szstp0     0.01      SZF - Initial step size
szerr      0.001     SZF - Error criterion
szstpmx    5.0       SZF - Maximum step size [=] m
szsz0      0.01      SZF - Initial Value of dellay*Ueff*Heff
!
!
!     /ALPHcom/
!
ialpfl     1.        ALPHI - calculation flag; 0) alpha=alpco; 1)1/(1+z); 2)1
alpco      0.2       ALPHI - Value for alpha if IALPFL = 0
!
!
!     /PHIcom/
!
iphifl     3.        PHIF - calc flag
dellay     2.15      Raito of Hl/Heff
!
!
!
!     /VUcom/
!
vua        1.3       Constant Av in source model
vub        1.2       Constant Bv in source model
vuc        20.0      Constant Ev in source model
vud        .64       Constant Dv in source model
vudelta    0.20      Constant DELTAv in source model
!
! End-of-File
```

Figure C.4. (concluded)

Pseudosteady-State Module--DEGADIS2

DEGADIS2 performs the downwind dispersion portion of the calculation for each of several observers released successively over the transient source described by DEGADIS1. (Note that the routines INCGAMMA, GAMMA, and SERIES are linked in DEGADIS2 but never are called in DEGADIS2.) DEGADIS2 is composed of the following subroutines (Figure C.5):

(*) AFGEN	is a utility which linearly interpolates between a pair of points based on a list of supplied values (F-3).
(*) AFGEN2	is a utility which linearly interpolates between a pair of points based on a list of supplied values (F-4).
(*) DEGADIS2	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS1, recover the numerical parameter file RUN_NAME.ER2, and perform the simulation (F-23).
(*) ESTRT2	recovers the numerical parameters contained in the file RUN_NAME.ER2, particularly the number of observers NOBS (F-53).
(*) OB	contains the ordinary differential equations which average the gas source for each observer (F-95).
(*) PSS	contains the ordinary differential equations which describe the portion of the downwind dispersion calculation when b > 0 (F-99).
(*) PSSOUT	governs the output of calculated points to the file RUN_NAME.PSD when PSS is active (F-102).

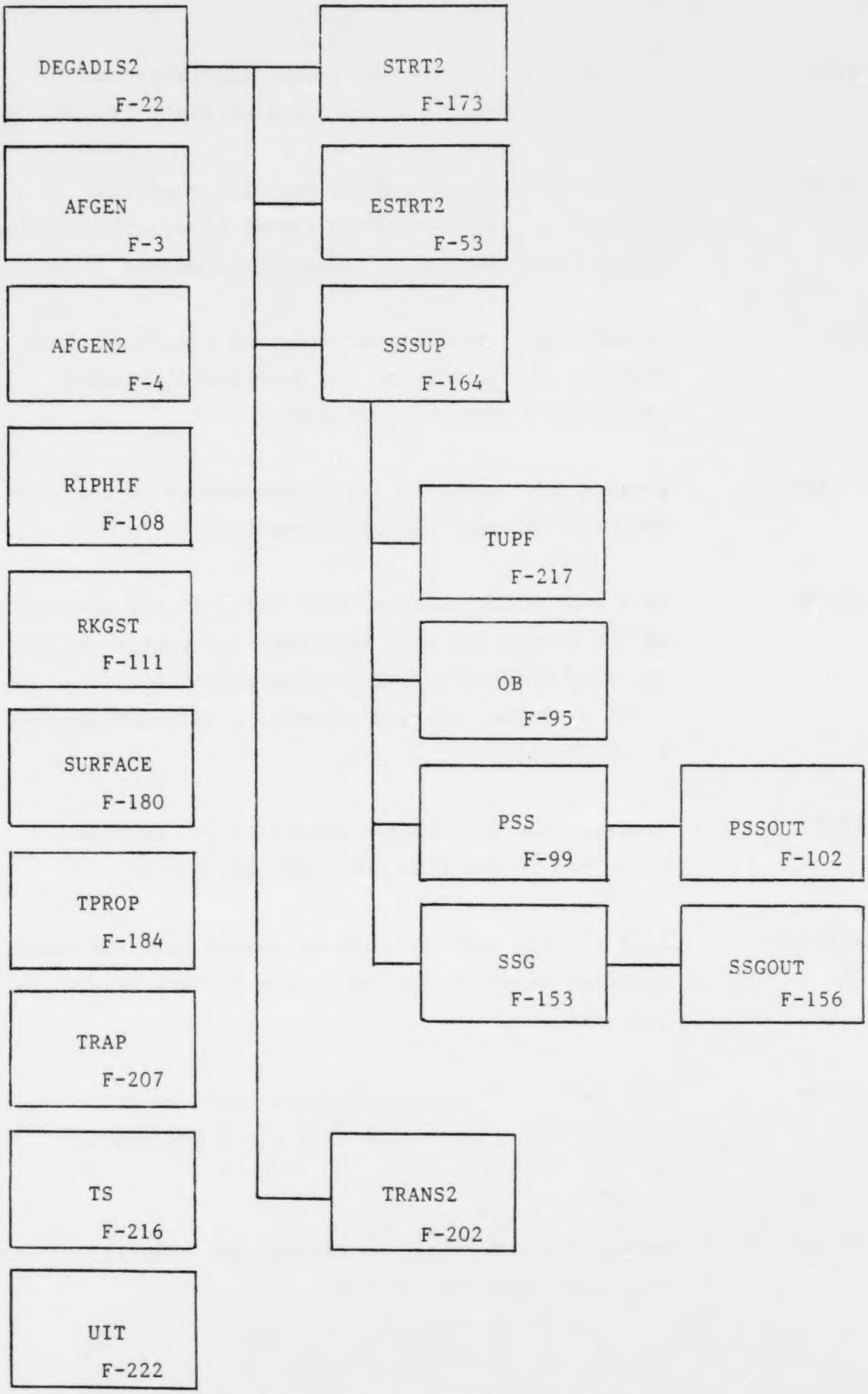

DEGADIS2
F-22
STRT2
F-173
AFGEN
F-3
ESTRT2
F-53
AFGEN2
F-4
SSSUP
F-164
RIPHIF
F-108
TUPF
F-217
RKGST
F-111
OB
F-95
SURFACE
F-180
PSS
F-99
PSSOUT
F-102
TPROP
F-184
SSG
F-153
SSGOUT
F-156
TRAP
F-207
TS
F-216
TRANS2
F-202
UIT
F-222

Figure C.5. DEGADIS2 flowchart.

(*) RIPHIF	is a series of utilities which calculates the Richardson number and the value of $\phi(Ri)$ (F-108).
(*) RKGST	is a utility routine which performs numerical integration of a specified system of equations using a modified fourth-order Runge-Kutta method (F-111).
(*) SSG	contains the ordinary differential equations which describe the portion of the downwind dispersion calculation when $b = 0$ (F-153).
(*) SSGOUT	governs the output of calculated points to the file RUN_NAME.PSD when SSG is active (F-156).
(*) SSSUP	is a supervisor routine which controls the averaging of the source for each observer, the portion of the downwind dispersion calculation when $b > 0$, and the portion of the downwind dispersion calculation when $b = 0$ (F-164).
(*) STRT2	recovers the information generated in DEGADIS1 contained in the file RUN_NAME.TR2 (F-173).
(*) SURFACE	is a utility routine which estimates heat and water transfer rates across the bottom surface of the gas layer (F-180).
(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (F-184).
(*) TRANS2	writes the information necessary for DEGADIS3 to the file (RUN_NAME.TR3) (F-202).

(*) TRAP	is a utility included for program diagnostics (F-207).
(*) TS	calculates the time when a given observer will be at a given downwind distance (F-216).
(*) TUPF	contains the two routines which determine the intersection of the upwind/downwind edge of the secondary gas source with a given observer (F-217).
(*) UIT	is a series of routines to calculate observer position and velocity as a function of time (F-222).

As input, DEGADIS2 requires two files:

(*) RUN_NAME.ER2	contains the various numerical parameters, particularly the number of observers NOBS. For most simulations, a copy of the SYS$DEGADIS:EXAMPLE.ER2 file will be adequate. (See Figure II.1 or II.2.) A copy of SYS$DEGADIS:EXAMPLE.ER2 is included in Figure C.6.
(*) RUN_NAME.TR2	contains the basic simulation definition as well as calculated secondary source parameters.

DEGADIS2 generates the following output files:

(*) RUN_NAME.OBS	contains a summary of the source parameters for each observer.

```
  ! This is an example for an 'ER2' run parameter file.
  ! The same rules apply as for the 'ER1' files.
  !
  !234567890123456789012345678901234567890
  !---------1---------2---------3
  !
  ! These values are in common area /ERROR/
  !
* SYOER       0.0      SSSUP - RKGST - INITIAL SY
  ERRO        0.005    SSSUP - RKGST(OBS) - ERROR BOUND
  SZOER       0.01     SSSUP - RKGST(OBS) - INITIAL SZ
  WTAIO       1.0      SSSUP - RKGST(OBS) - WEIGHT FOR AI
  WTQOO       1.0      SSSUP - RKGST(OBS) - WEIGHT FOR Q
  WTSZO       1.0      SSSUP - RKGST(OBS) - WEIGHT FOR SZ
* ERRP        0.003    SSSUP - RKGST(PSS) - ERROR BOUND
* SMXP        10.      SSSUP - RKGST(PSS) - MAXIMUM STEP
* WTSZP       1.0      SSSUP - RKGST(PSS) - WEIGHT FOR SZ
* WTSYP       1.0      SSSUP - RKGST(PSS) - WEIGHT FOR SY
* WTBEP       1.0      SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
* WTDH        1.0      SSSUP - RKGST(PSS) - WEIGHT FOR DH
* ERRG        0.003    SSSUP - RKGST(SSG) - ERROR BOUND
* SMXG        10.      SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
  ERTDNF      0.0005   TDNF - CONVERGENCE CRITERION
  ERTUPF      0.0005   TUPF - CONVERGENCE CRITERION
* WTRUH       1.0      SSSUP - RKGST(SSG) - WEIGHT FOR RUH
* WTDHG       1.0      SSSUP - RKGST(SSG) - WEIGHT FOR DH
  !
  ! These values are in common area /STP/
  !
  STPO        0.05     SSSUP - RKGST(OBS) - INITIAL STEP
* STPP        0.05     SSSUP - RKGST(PSS) - INITIAL STEP
* ODLP        0.06     SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
* ODLLP       80.      SSSUP-RKGST(PSS)-MAXIMUM DISTANCE BETWEEN OUTPUTS(m)
* STPG        0.05     SSSUP - RKGST(SSG) - INITIAL STEP
* ODLG        0.045    SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
* ODLLG       80.      SSSUP-RKGST(SSG)-MAXIMUM DISTANCE BETWEEN OUTPUTS(m)
  !
  ! The last variable NOBS is in /CNOBS/
  !
  ! Note: it is read in as a real value even though it is integer type
  !      in the program.
  !
  NOBS        30.
  !
  !
  !
  ! End-of-File
```

*used by steady-state simulation

Figure C.6. SYS$DEGADIS:EXAMPLE.ER2 listing.

(*) RUN_NAME.PSD	contains the calculated downwind dispersion parameters for each observer. DEGADIS3 and DEGADIS4 sort this information to determine the downwind concentration profiles as a function of position and time.
(*) RUN_NAME.TR3	contains the simulation definition and the number of each record type written to RUN_NAME.PSD.

Time Sort Modules--DEGADIS3 and DEGADIS4

DEGADIS3 sorts the downwind dispersion calculation for each of several observers for concentration information at several given times; the along-wind dispersion correction is then applied as desired. DEGADIS3 uses the following subroutines (Figure C.7).

(*) DEGADIS3	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS2, recover the numerical parameter file RUN_NAME.ER3, sort and apply the along-wind dispersion correction to the results of DEGADIS2, and output the results (F-28).
(*) ESTRT3	recovers the numerical parameters contained in the file RUN_NAME.ER3, particularly the time sort parameters (F-57).
(*) GAMMA	is a utility function that calculates the gamma function of the argument (i.e. $\Gamma(x)$) (F-58).
(*) GETTIM	sets the default time sort parameters as needed (F-60).
(*) INCGAMMA	is a utility function that calculates the incomplete gamma function of the two arguments (F-69).

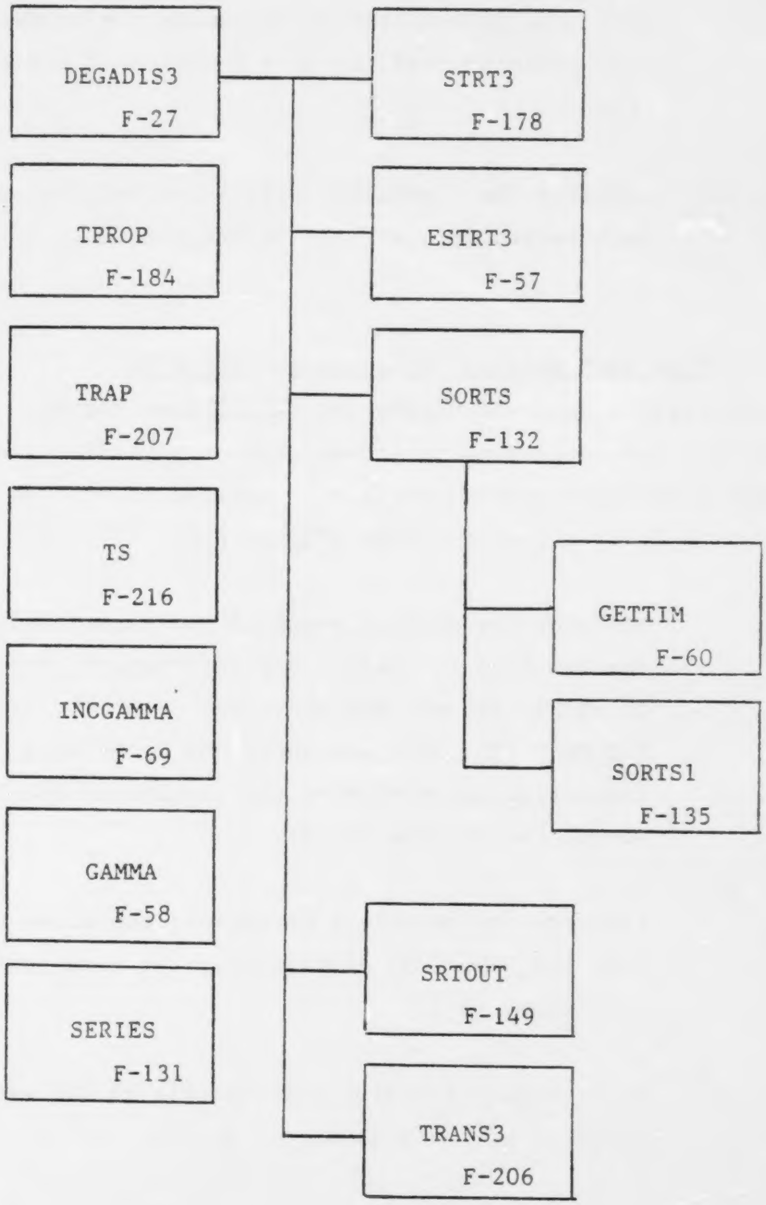

DEGADIS3
F-27
TPROP
F-184
TRAP
F-207
TS
F-216
INCGAMMA
F-69
GAMMA
F-58
SERIES
F-131
STRT3
F-178
ESTRT3
F-57
SORTS
F-132
GETTIM
F-60
SORTS1
F-135
SRTOUT
F-149
TRANS3
F-206

Figure C.7. DEGADIS3 flowchart.

(*) SERIES	evaluates a series needed to estimate the mass of gas above a given concentration level (F-131).
(*) SORTS	recovers the information in RUN_NAME.PSD and arranges the information according to the time sort parameters in the file RUN_NAME.ER3 (F-132).
(*) SORTS1	applies the along-wind dispersion correction to the time-sorted information (F-135).
(*) SRTOUT	generates the formatted output file RUN_NAME.SR3 (F-149).
(*) STRT3	recovers the information generated in DEGADIS2 contained in the file RUN_NAME.TR3 (F-178).
(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (F-184).
(*) TRANS3	writes RUN_NAME.TR4 which contains the necessary information to recover the other output files for this simulation (F-206).
(*) TRAP	is a utility included for program diagnostics (F-207).
(*) TS	calculates the time when a given observer will be at a given downwind distance (F-216).

As input, DEGADIS3 requires three files:

(*) RUN_NAME.ER3	contains various numerical parameters including the time sort parameters and the flag which dictates whether the along-wind dispersion correction is applied. A copy of SYS$DEGADIS:EXAMPLE.ER3 file uses the default time sort parameters and includes the along-wind dispersion correction which should apply for most simulations. (See Figure II.2.) A copy of SYS$DEGADIS:EXAMPLE.ER3 is included in Figure C.8.
(*) RUN_NAME.PSD	contains the calculated downwind dispersion parameters for each observer. DEGADIS3 sorts this information to determine the downwind concentration profiles as a function of position at a given time.
(*) RUN_NAME.TR3	contains the number of each record type written to RUN_NAME.PSD as well as the simulation definition.

As output, DEGADIS3 generates two new files:

(*) RUN_NAME.SR3	is the formatted output list of the time-sorted concentration parameters. Concentration contours are generated for the specified upper and lower flammability at the specified height entered in DEGADISIN or OOMS_IN. An example is included in Section IV.
(*) RUN_NAME.TR4	contains the necessary information to recover the other output files to facilitate further processing.

```
  ! This is an example for an 'ER3' run parameter file.
  ! The same rules apply as for the 'ER1' files.
  !
  !234567890123456789012345678890
  !--------1---------2---------3
  !
  ! These values are in common area /ERROR/
  !
  ERT1       20.       FIRST SORT TIME
  ERDT       5.        SORT TIME DELTA
  ERNTIM     20.       NUMBER OF TIMES FOR THE SORT
  !
  !    Note: ERNTIM is entered as a real variable even though
  !          it is an integer type variable in the program.
  !
  !  The value of CHECK5 determines whether the above sort parameters
  !     are used.  CHECK5 is initialized through the passed transfer
  !     files to .FALSE.  CHECK5 is set to .TRUE. if a real value of 1.
  !     is passed in this file.
  !
  CHECK5       0.       USE THE DEFAULT TIME PARAMETERS
  !CHECK5      1.       USE THE TIME PARAMETERS GIVEN ABOVE
  !
  !
  sigx_flag    1.       correction for x-direction dispersion is to be made
  !sigx_flag    0.       no correction for x-direction dispersion
  !
  !
  !
  ! End-of-File
####
```

Figure C.8. SYS$DEGADIS:EXAMPLE.ER3 listing.

DEGADIS4 sorts the downwind dispersion calculation for each of several observers for concentration information at several given positions; the along-wind dispersion correction is then applied as desired. DEGADIS4 uses the following subroutines (Figure C.9):

(*) DEGADIS4	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS2, recover the numerical parameter file RUN_NAME.ER3, sort and apply the along-wind dispersion correction to the results of DEGADIS2, and output the results (F-33).
(*) DOSOUT	generates the formatted output file RUN_NAME.SR4 (F-44).
(*) ESTRT3	recovers the numerical parameters contained in the file RUN_NAME.ER3, particularly whether the x-direction dispersion correction is to be applied (F-57).
(*) GAMMA	is a utility function that calculates the gamma function of the argument (i.e. $\Gamma(x)$) (F-58).
(*) GETTIMDOS	sets the time sort parameters as required to output the concentration time history at the desired positions (F-62).
(*) INCGAMMA	is a utility function that calculates the incomplete gamma function of the two arguments (F-69).
(*) SERIES	evaluates a series needed to estimate the mass of gas above a given concentration level (F-131).

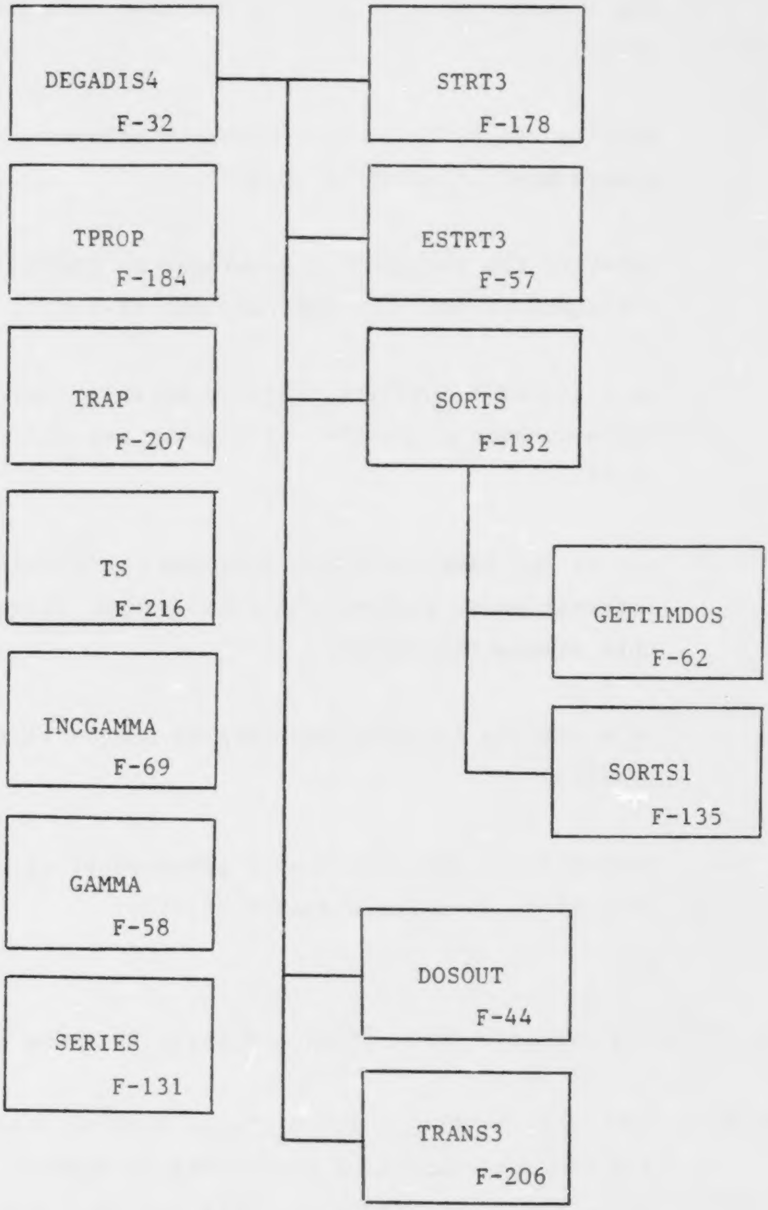

DEGADIS4
F-32
TPROP
F-184
TRAP
F-207
TS
F-216
INCGAMMA
F-69
GAMMA
F-58
SERIES
F-131
STRT3
F-178
ESTRT3
F-57
SORTS
F-132
GETTIMDOS
F-62
SORTS1
F-135
DOSOUT
F-44
TRANS3
F-206

Figure C.9. DEGADIS4 flowchart.

(*) SORTS	recovers the information in RUN_NAME.PSD and arranges the information according to the time sort parameters (F-132).
(*) SORTS1	applies the x-direction dispersion correction to the time-sorted information (F-135).
(*) STRT3	recovers the information generated in DEGADIS2 contained in the file RUN_NAME.TR3 (F-178).
(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (F-184).
(*) TRANS3	writes RUN_NAME.TR4 which contains the necessary information to recover the other output files for this simulation (F-206).
(*) TRAP	is a utility included for program diagnostics (F-207).
(*) TS	calculations the time when a given observer will be at a given downwind distance (F-216).

As input, DEGADIS4 requires three files and input from the terminal:

(*) RUN_NAME.ER3	contains the flag which dictates whether the x-direction dispersion correction is applied. A copy of SYS$DEGADIS:EXAMPLE.ER3 file includes the x-direction dispersion correction which should apply for most simulations. (See Figure II.2.) A copy of SYS$DEGADIS:EXAMPLE.ER3 is included in Figure C.8.

(*) RUN_NAME.PSD	contains the calculated downwind dispersion parameters for each observer. DEGADIS4 sorts this information to determine the downwind concentration time histories at the desired positions.
(*) RUN_NAME.TR3	contains the number of each record type written to RUN_NAME.PSD as well as the simulation definition.
(*) terminal input	DEGADIS4 prompts the user for the file name to be used for this run. In addition, DEGADIS4 requests the number of downwind distances (JDOS). For each downwind distance, DEGADIS4 asks for the x-position (DOSDISX(I)). Four positions are allowed for each downwind distance (DOSDISY(IJ,I) and DOSDISZ(IJ,I) for IJ=1 to 4). If fewer than four positions are desired, negative values are entered for the first position which is not desired. A summary of this input information is included in Figure C.10. Note that the same information can be put in a command file for batch processing. As well, the same information can be put in a file which can be associated with FOR005.DAT so that a file can be used for input.

```
                         RUN_NAME
                         JDOS
                     (   DOSDISX(I)
                     (  (DOSDISY(1,I),DOSDISZ(1,I)
FOR I=1 TO JDOS     {   (DOSDISY(2,I),DOSDISZ(2,I)
                     ( *{DOSDISY(3,I),DOSDISZ(3,I)
                     (  (DOSDISY(4,I),DOSDISZ(4,I)
```

*Note that if fewer than four positions are desired, negative values are entered for the first position which is not desired.

Figure C.10. Structure of input for DEGADIS.

As output, DEGADIS4 generates two new files:

(*) RUN_NAME.SR4	is the formatted output list of the sorted concentration time histories. An example is included in Section IV.
(*) RUN_NAME.TR4	contains the necessary information to recover the other output files to facilitate further processing.

Steady-State Module--SDEGADIS2

SDEGADIS2 is a simplification of DEGADIS2 which uses many of the same subroutines. SDEGADIS2 performs the downwind dispersion portion of the calculation for a steady-state source described by DEGADIS1. SDEGADIS2 is composed of the following subroutines (Figure C.11):

(*) AFGEN	is a utility which linearly interpolates between a pair of points based on a list of supplied values (F-3).
(*) GAMMA	is a utility function that calculates the gamma function of the argument (i.e. $\Gamma(x)$) (F-58).
(*) ESTRT2SS	recovers a subset of the numerical parameters contained in the file RUN_NAME.ER2 as indicated in Figure C.6 (F-55).
(*) INCGAMMA	is a utility function that calculates the incomplete gamma function of the two arguments (F-69).
(*) PSS	is the same subroutine used in DEGADIS2; it contains the ordinary differential equations which describe the downwind dispersion calculation when $b > 0$ (F-99).

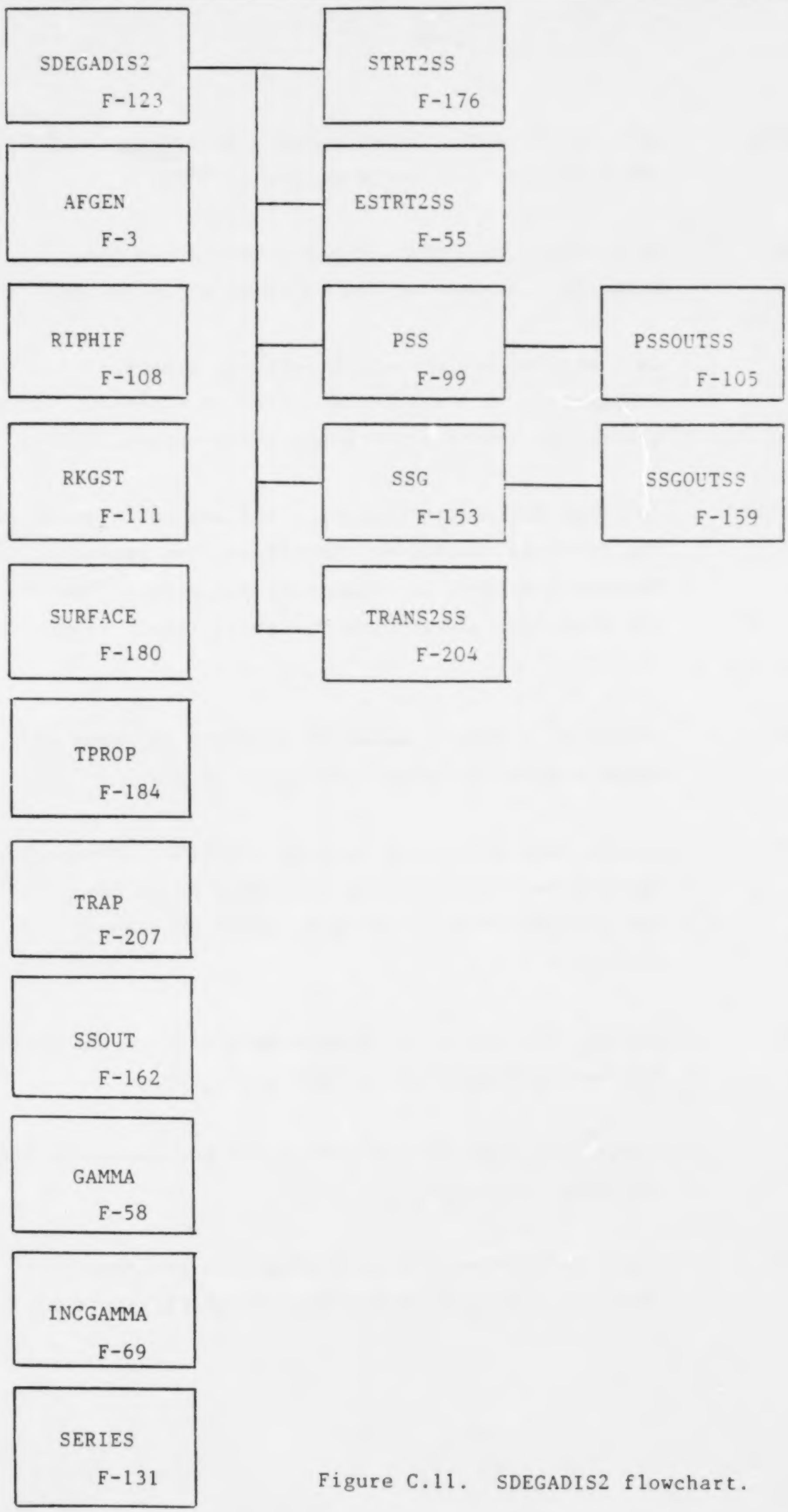

SDEGADIS2
F-123
AFGEN
F-3
RIPHIF
F-108
RKGST
F-111
SURFACE
F-180
TPROP
F-184
TRAP
F-207
SSOUT
F-162
GAMMA
F-58
INCGAMMA
F-69
SERIES
F-131
STRT2SS
F-176
ESTRT2SS
F-55
PSS
F-99
SSG
F-153
TRANS2SS
F-204
PSSOUTSS
F-105
SSGOUTSS
F-159

Figure C.11. SDEGADIS2 flowchart.

(*) PSSOUTSS	governs the output of calculated points to the file RUN_NAME.SR3 when PSS is active (F-105).
(*) RIPHIF	is a series of utilities which calculates the Richardson number and the value of ϕ(Ri) (F-108).
(*) RKGST	is a utility routine which performs numerical integration of a specified system of equations using a modified fourth-order Runge-Kutta method (F-111).
(*) SDEGADIS2	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS1, recover the numerical parameters file RUN_NAME.ER2, and perform the steady-state simulation (F-123).
(*) SERIES	evaluates a series needed to estimate the mass of gas above a given concentration level (F-131).
(*) SSG	is the same subroutine used in DEGADIS2; it contains the ordinary differential equations which describe the downwind dispersion calculation when b = 0 (F-153).
(*) SSGOUT	governs the output of calculated points to the file RUN_NAME.SR3 when SSG is active (F-156).
(*) SSOUT	writes RUN_NAME.SR3 and calculates the concentration contours (F-162).
(*) STRT2SS	recovers a subset of the information generated in DEGADIS1 contained in the file RUN_NAME.TR2 (F-176).

(*) SURFACE	is a utility routine which estimates heat and water transfer rates across the bottom surface of the gas layer (F-180).
(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (F-184).
(*) TRANS2SS	writes RUN_NAME.TR3 (F-204).
(*) TRAP	is a utility included for program diagnostics (F-207).

As input, SDEGADIS2 requires two files:

(*) RUN_NAME.ER2	contains various numerical parameters; the steady-state simulation requires only part of these. For most simulations, a copy of the SYS$DEGADIS:EXAMPLE.ER2 file will be adequate. See Figure II.1. A copy of SYS$DEGADIS:EXAMPLE.ER2 is included in Figure C.6.
(*) RUN_NAME.TR2	contains the basic simulation definition as well as calculated secondary source parameters; the steady-state simulation requires only part of these.

As output, SDEGADIS2 generates the following files:

(*) RUN_NAME.SR3	is the formatted output list of the downwind concentration parameters. Concentration contours are generated for the specified upper and lower flammability at the specified height entered in DEGADISIN.
(*) RUN_NAME.TR3	contains the necessary information to recover the other output files to facilitate further processing.

APPENDIX D
DEGADIS EXAMPLE SIMULATION

In 1980, the U.S. Department of Energy sponsored at China Lake, California, a series of nine LNG releases referred to as the Burro series of experiments (Koopman et al., 1982). The release condition (Table D.1) for the numerical examples in this Appendix are those of Burro 9 which was modeled both as a steady-state and transient (time-limited) release. As suggested by the Shell Maplin Sands LNG releases (Blackmore et al., 1982), the liquid source diameter was determined using a boiling rate of 0.085 kg/m^2 for LNG on water.

TABLE D.1
SUMMARY OF BURRO 9 TEST CONDITIONS
USED IN EXAMPLE SIMULATIONS

Source Rate:	130.0 kg/s
Source Radius:	22.06 m
Wind Speed:	6.5 m/s at 8.0 m
Atmospheric Stability:	C (Pasquill)
Monin-Obukhov Length:	-140. m
Surface Roughness:	2.05×10^{-4} m
Air Temperature:	35.4°C
Atmospheric Humidity:	12.5%
Surface Temperature:	310K

Example Input Sessions

The input procedures for simulation of the transient release (RUN_NAME=BURRO9) and the steady-state release (RUN_NAME=BURRO9S) are very similar. Therefore, only the specification of the source rate and extent have been included for the transient release. In the point-by-point discussion of the input procedure, note the following:

(*) A line terminator (normally a carriage return) must end every line entered by the user.

(*) The file name specification RUN_NAME must satisfy system restrictions.

(*) When DEGADISIN requests the user to choose an option, all acceptable responses are a single character (capital or lower case). The default responses are denoted by a capital letter inside angle brackets (e.g. <N>). When applicable, a menu of responses is included inside parentheses.

(*) For numerical responses, a comma, space, tab, or line terminator (carriage return) may separate the numbers.

(*) When a file is used as input (i.e. for the density or transient source input), DEGADISIN reads the same information from the file which would be entered at the terminal in the same order and in the same format.

Preceding page blank

Notes on Steady-State Simulation of BURRO9

① Begin the input procedure by execution of DEGADISIN.

② The file name specification must follow system restrictions. The DEGADIS model uses this file name along with various file extensions for input and output.

③ The Title Block is used to carry any desired comments such as information on the specification of certain parameters.

④ The wind field parameters include the wind velocity (m/s) at a specified height (m) and the surface roughness (m).

⑤ The Pasquill stability class is used to generate estimates of other atmospheric parameters which follow.

⑥ The averaging time is used to determine the value of δ (DELTA) in the lateral dispersion coefficient specification. Changes to the values of δ are calculated assuming that the effect of averaging time only influences the lateral plume meander of a steady-state release. Note that DEGADIS does not currently evaluate time-averaged concentrations for time-limited releases.

⑦ The current settings of pertinent atmospheric parameters are displayed in this list. If any of these are to be changed, the first letter of the parameter to be changed is entered. Note that the default--indicated by <N>--is No for no changes.

⑧ The Monin-Obukhov length (Length in the list) is to be changed, so L is entered in response to the prompt.

⑨ The list is redisplayed to verify the change and to request any further changes. The (default) response of No causes the program to go to the next question.

⑩ The ambient temperature and pressure are entered.

⑪ DEGADISIN calculates the ambient air density for the given input parameters.

```
①  $ RUN SYS$DEGADIS:DEGADISIN

                     DEnse GAs DISpersion Model input module.

②  Enter the simulation name : [DIR]RUNNAME BURRO9S
       INPUT MODULE -- DEGADIS MODEL

       **********************************
③      Enter Title Block -- up to 4 lines of 80 characters
       To stop, type '//'
   Steady-state simulation of BURRO 9
   //
④      ENTER WIND PARAMETERS -- U0 (m/s), Z0 (m), and ZR(m)
       U0 -- Wind velocity at reference height Z0
       ZR -- Surface Roughness
   6.5,8.,2.05E-4

⑤      Enter the Pasquill stability class: (A,B,C,D,E,F) <D> C
⑥  Enter the averaging time (s) for estimating DELTA: 0.

⑦  The values for the atmospheric parameters are set as follows:
   DELTA:                          0.1090
   BETA:                           0.8940
   Monin-Obukhov length:          -9.3344 m
   Sigma X Coefficient:            0.0200
   Sigma X Power:                  1.2200
   Sigma X Minimum Distance:     130.0000 m
   Do you wish to change any of these?
⑧  (No,Delta,Beta,Length,Coefficient,Power,Minimum) <N> L
   Note: For infinity, ML = 0.0
   Enter the desired Monin-Obukhov length: (m) -140.

⑨  The values for the atmospheric parameters are set as follows:
   DELTA:                          0.1090
   BETA:                           0.8940
   Monin-Obukhov length:        -140.0000 m
   Sigma X Coefficient:            0.0200
   Sigma X Power:                  1.2200
   Sigma X Minimum Distance:     130.0000 m
   Do you wish to change any of these?
   (No,Delta,Beta,Length,Coefficient,Power,Minimum) <N>

⑩  Enter the ambient temperature(C) and pressure(atm): 35.4,0.94

   The ambient humidity can be entered as Relative or Absolute.
   Enter either R or A <R or a>:
   Enter the relative humidity (%): 12.5

⑪  Ambient Air density is    1.0720     kg/m**3
```

⑫ If the release is isothermal, respond "Y". A positive response causes DEGADISIN to ask for a list of concentration, density, and mole fraction points for the gas mixture. The default response is negative.

⑬ If the release is simulated as adiabatic, the default negative response is chosen. For inclusion of heat transfer effects, the surface temperature and the method of calculating the heat transfer coefficient must be specified.

⑭ Water transfer to the source blanket (if present) can be included in the calculation.

⑮ Enter the three-letter designation of the diffusing gas. The properties of LNG as methane, LPG as propane, and NH_3 (ammonia) are included among others.

⑯ A list of the properties for the specified gas (if available) is given. If any of the parameters are to be changed, the first letter of the parameter to be changed in the list is given to the prompt. Here, the level at which the flammability contours are calculated is changed from 0.5 m to 1.0 m.

⑰ The gas property list is displayed again. The default response is no change.

⑱ The lowest concentration of interest is the concentration at which the calculations are stopped.

```
⑫  Is this an Isothermal spill? <y or N>

⑬  Is heat transfer to be included in the calculations <y or N> Y
    Enter the surface temperature [=] K : 310.
    Do you want to use the built in correlation, the LLNL correlation, or
    enter a particular value?
    (Corr,LLNLcorr,Value)  <C>

⑭  Is water transfer to be included in the source <y or N>

⑮  Enter the code name of the diffusing species: LNG

⑯  The characteristics for the gas are set as follows:
    Molecular weight:                                  16.04
    Storage temperature [K]:                            111.70
    Density at storage temperature, PAMB  [kg/m**3]:    1.6845
    Mean Heat capacity constant                        5.60000E-08
    Mean Heat capacity power                            5.0000
    Upper Flammability Limit [mole frac]                0.15000
    Lower Flammability Limit [mole frac]                5.00000E-02
    Height of Flammability Limit [m]                    0.50000
    Do you wish to change any of these? (No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z) <N> Z
    Enter the desired Height for the flammable limit calculations: 1.

⑰  The characteristics for the gas are set as follows:
    Molecular weight:                                  16.04
    Storage temperature [K]:                            111.70
    Density at storage temperature, PAMB  [kg/m**3]:    1.6845
    Mean Heat capacity constant                        5.60000E-08
    Mean Heat capacity power                            5.0000
    Upper Flammability Limit [mole frac]                0.15000
    Lower Flammability Limit [mole frac]                5.00000E-02
    Height of Flammability Limit [m]                    1.0000
    Do you wish to change any of these? (No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z) <N>

⑱      The suggested LOWEST CONCENTRATION OF INTEREST (gas_lfl/2.)
        is   1.52452E-02 kg/m**3.  Enter the desired value: 0.015
```

⑲ The BURRO9 case is for a release of "pure" (undiluted) LNG. For diluted releases, DEGADIS requires the mass fraction of the contaminant and the mixture temperature.

⑳ If a steady-state release is to be simulated, type "Y" to the prompt. For a steady simulation, the steady-state mass evolution rate (kg/s) and primary source extent (m) are required.

㉑ A note about the numerical parameter files is included. These files contain various constant values used in the programs to which the user has access without recompiling the programs. Access is granted as a convenience.

㉒ DEGADISIN will generate a command procedure suitable for running the model under VMS.

㉓ If so desired, DEGADISIN will initiate the command procedure under VMS. If not, the program returns to the operating system.

```
     Specification of source parameters.

⑲   Is this a release of pure (P) or diluted (d) material specified above? <P or d>

⑳   Is this a Steady state simulation? <y or N> Y

     Enter the desired evolution rate [=] kg/sec : 130.
     Enter the desired source radius [=] m : 22.06

㉑   In addition to the information just obtained, DEGADIS
     requires a series of numerical parameter files which use
     the same name as [DIR]RUNNAME given above.

        For convenience, example parameter files are included for
     each step.  They are:
             EXAMPLE.ER1 and
             EXAMPLE.ER2
      Note that each of these files can be edited during the course of the
     simulation if a parameter proves to be out of specification.

㉒   Do you want a command file to be generated to execute the procedure? <Y or n>
     The command file will be generated under the file name:
             BURRO9S.com

㉓   Do you wish to initiate this procedure?  <y or N>
     $
```

Notes on Transient Simulation of BURRO9

Beginning with the specification of the source rate and extent, the responses to all of the previous questions except the simulation name (RUN_NAME) are the same for the steady-state case and are not repeated.

⑲ The BURRO9 case is for a release of "pure" (undiluted) LNG. For diluted releases, DEGADIS requires the mass fraction of the contaminant and the mixture temperature.

⑳ The default response is for a transient release.

㉑ An initial mass of gas can be specified over the source. This can be used to model aboveground releases such as the Thorney Island Trials.

㉒ The transient source description consists of ordered triples of time, evolution rate, and source radius for pure sources. For diluted sources, values of evolution rate, source radius, contaminant mass fraction, and source temperature must be specified as functions of time.

㉓ An input file can be used to enter the data triples to avoid typing errors or to use as output from another model such as a liquid spreading model. The file format is the same as the terminal entry format.

㉔ The first item is the number of triples used in the description followed by the triples with the last two values showing no gas present.

㉕ A note about the numerical parameter files is included. These files contain various constant values used in the programs to which the user has access without recompiling the programs. Access is granted as a convenience.

㉖ DEGADISIN will generate a command procedure suitable for running the model under VMS.

㉗ If so desired, DEGADISIN will initiate the procedure under VMS. If not, the program returns to the operating system.

```
Specification of source parameters.

⑲  Is this a release of pure (P) or diluted (d) material specified above? <P or d>

⑳  Is this a Steady state simulation? <y or N>

㉑  Enter the initial mass of pure gas over the source. (kg)
    (Positive or zero): 0.

㉒            Source Description

       The description of the primary source mass
       evolution rate E and radius R1 for a transient
       release is input by ordered triples as follows:

          first point        -- t=0, E(t=0), R1(t=0) (initial, nonzero values)
          second point       -- t=t1, E(t=t1), R1(t=t1)
                 .
                 .
                 .

          last nonzero point -- t=TEND, E(t=TEND), R1(t=TEND)
          next to last point -- t=TEND+1., E=0., R1=0.
          last point         -- t=TEND+2., E=0., R1=0.

       Note: the final time (TEND) is the last time when E and R1 are non-zero.

㉓  Do you have an input file for the Source Description? [y or N]

       Enter the number of triples (max= 30) starting with t=0. and ending
       with t=TEND+2. for the source description: 4

㉔     Enter TIME (sec), EVOLUTION RATE (kg/s), and POOL RADIUS (m)
       0.,130.,22.06
       80.,130.,22.06
       81.,0.,0.
       82.,0.,0.

㉕     In addition to the information just obtained, DEGADIS
       requires a series of numerical parameter files which use
       the same name as [DIR]RUNNAME given above.

         For convenience, example parameter files are included for
       each step.  They are:
               EXAMPLE.ER1,
               EXAMPLE.ER2, and
               EXAMPLE.ER3
        Note that each of these files can be edited during the course of the
       simulation if a parameter proves to be out of specification.

㉖     Do you want a command file to be generated to execute the procedure? <Y or n>
       The command file will be generated under the file name:
               BURRO9.com

㉗     Do you wish to initiate this procedure?  <y or N>
       $
```

The generated INP files for BURRO9S and BURRO9 are shown in Figures IV.1 and IV.2. If necessary, the user may edit the INP file before beginning the simulation. The generated command procedures are shown in Figures D.1 and D.2.

Example Simulation Output

After proper completion of the model, BURRO9.LIS and BURRO9S.LIS contain the output listing for the transient and steady-state releases, respectively. A discussion of the steady-state and transient simulation listings follows. Because of the similarities between the steady-state and transient simulation listings, the first portion of the transient simulation is not included.

```
Steady-state siaulation of BURRO 9


  6.500000        8.000000        2.0500000E-04
    3
 0.1090000       0.8940000       -140.0000
 2.0000000E-02    1.220000        130.0000
  308.5500       0.9400000       5.1530502E-03
   0   310.0000
   1  0.0000000E+00
   0  0.0000000E+00
LNG
   16.04000       111.7000        1.684480
  5.6000000E-08   5.000000
  0.1500000      5.0000000E-02    1.000000
  1.5000000E-02
  0.0000000E+00
     4
  0.0000000E+00   130.0000        22.06000        1.000000        111.7000        1.000000
   6023.000       130.0000        22.06000        1.000000        111.7000        1.000000
   6024.000      0.0000000E+00   0.0000000E+00    1.000000        111.7000        1.000000
   6025.000      0.0000000E+00   0.0000000E+00    1.000000        111.7000        1.000000
 F F F F T F
 20-OCT-1987 14:09:42.08
   130.0000       44.12000        17.32588
```

Figure D.1. BURRO9S.INP listing.

```
Time-limited simulation of BURRO 9



   6.500000       8.000000       2.0500000E-04
     3
  0.1090000      0.8940000      -140.0000
  2.0000000E-02   1.220000       130.0000
   308.5500      0.9400000      5.1530502E-03
    0   310.0000
    1  0.0000000E+00
    0  0.0000000E+00
LNG
   16.04000       111.7000       1.684480
  5.6000000E-08   5.000000
  0.1500000      5.0000000E-02   1.000000
  1.5000000E-02
  0.0000000E+00
     4
  0.0000000E+00   130.0000       22.06000       1.000000       111.7000       1.000000
   80.00000       130.0000       22.06000       1.000000       111.7000       1.000000
   81.00000      0.0000000E+00  0.0000000E+00   1.000000       111.7000       1.000000
   82.00000      0.0000000E+00  0.0000000E+00   1.000000       111.7000       1.000000
 F F F F F F
20-OCT-1987 14:13:26.51
```

Figure D.2. BURRO9.INP listing.

Notes on Steady-State Simulation of BURRO9

① The date and time DEGADISIN was run are included.

② The input information gathered by DEGADISIN is repeated to assist in documentation of the simulations. Included here are the Title Block and the atmospheric conditions.

```
**********************      U O A _ D E G A D I S   M O D E L   O U T P U T   - -   V E R S I O N   2.0      ********

****************                    **************** 20-OCT-1987 14:14:46.01 ****************                    ***

①   Data input on                 20-OCT-1987 14:09:42.08
    Source program run on         20-OCT-1987 14:14:46.01


            ****************************************************************************************************
            *                                                                                                  *
            *      NOTE:                                                                                       *
            *      -----                                                                                       *
            *                                                                                                  *
            *      >    All Calculations are limited to circular liquid sources.                               *
            *                                                                                                  *
            ****************************************************************************************************


②           TITLE BLOCK

    Steady-state simulation of BURRO 9



        Wind velocity at reference height                      6.50  m/s
        Reference height                                       8.00  m

        Surface roughness length                          2.050E-04  m

        Pasquill Stability class                               C

        Monin-Obukhov length                              -140.      m
        Gaussian distribution constants      Delta          0.10900  m
                                             Beta           0.89400

        Wind velocity power law constant     Alpha          0.10632
        Friction velocity                                   0.21878  m/s

        Ambient Temperature                                  308.55  K

        Surface Temperature                                  310.00  K
        Ambient Pressure                                      0.940  atm
        Ambient Absolute Humidity                         5.153E-03  kg/kg BDA
        Ambient Relative Humidity                             12.50  %

        Adiabatic Mixing:  Mole fraction  CONCENTRATION OF C     GAS DENSITY        Enthalpy          Temperature
                                                kg/m**3            kg/m**3            J/kg                K
                              0.00000           0.00000            1.07203          0.00000E+00         308.55
                              0.00896           0.00538            1.07502          -2043.4             306.53
                              0.01786           0.01078            1.07800          -4086.7             304.53

                              0.02668           0.01621            1.08097          -6130.1             302.56
                              0.04411           0.02717            1.08689          -10217.             298.67
                              0.06128           0.03825            1.09279          -14304.             294.87
```

③ Continuing with the input information, the contaminant gas properties are output.

④ The specification of the mass evolution rate, source radius, contaminant mass fraction, and source temperature are output. For a steady-state release, there is no initial mass in the cloud, and the source parameters are held constant for an arbitrarily large period of time.

⑤ Finally, certain numerical parameters and calculation flags are displayed. Some of these are set in DEGADISIN while others are set in the numerical parameter files.

```
          0.08653        0.05508        1.10163       -20434.          289.31
          0.11121        0.07217        1.11034       -26564.          283.31
          0.14325        0.09537        1.12198       -34737.          276.57

          0.17435        0.11903        1.13358       -42911.          270.29
          0.22661        0.16005        1.14325       -57214.          262.00
          0.26935        0.19598        1.15282       -69474.          254.87

                                           •

                                           •

                                           •

          0.88801        1.25418        1.53887    -3.33068E+05        132.99
          0.92334        1.37336        1.57857    -3.55545E+05        126.35
          0.96575        1.53473        1.63269    -3.84152E+05        118.34

          1.00000        1.68448        1.68448    -4.08672E+05        111.70
```

```
③      Specified Gas Properties:

          Molecular weight:                                      16.040
          Storage temperature:                                   111.70     K
          Density at storage temperature and ambient pressure:    1.6845     kg/m**3
          Mean heat capacity constant:                          5.60000E-08
          Mean heat capacity power:                              5.0000
          Upper mole fraction contour:                          0.15000
          Lower mole fraction contour:                          5.00000E-02
          Height for isopleths:                                  1.0000     m


④    Source input data points

              Initial mass in cloud:    0.00000E+00

       Time         Contaminant     Source Radius    Contaminant       Temperature        Enthalpy
                     Mass Rate                       Mass Fraction
         s            kg/s               m         kg contam/kg mix        K                J/kg
   0.00000E+00       130.00           22.060           1.0000            111.70          -4.08672E+05
     6023.0          130.00           22.060           1.0000            111.70          -4.08672E+05
     6024.0        0.00000E+00      0.00000E+00        1.0000            111.70          -4.08672E+05
     6025.0        0.00000E+00      0.00000E+00        1.0000            111.70          -4.08672E+05


⑤      Calculation procedure for ALPHA:  1

        Entrainment prescription for PHI:  3

        Layer thickness ratio used for average depth:    2.1500

        Air entrainment coefficient used: 0.590
```

⑥ A summary of the calculated secondary source parameters is included. The secondary source gas radius and height are output as functions of time along with other secondary parameters including the source mass flux (Qstar), the vertical concentration distribution parameter at the downwind edge (SZ(x = L/2.)), the contaminant mole fraction (Mole frac C), the gas mixture density (Density), and the Richardson number based on the cloud spreading velocity (Rich No.).

⑦ For a steady-state release, the source calculations are terminated after the calculated parameters are no longer changing as a function of time. A summary of the steady-state secondary source is included.

⑧ The downwind portion of the calculations is included. The distance downwind of the source is given in the first column. Columns 2 through 6 contain the mole fraction, contaminant concentration, mixture density, ratio of $(\rho - \rho_a)/c_c$ (Gamma), and mixture temperature on the centerline of the gas cloud at ground level. Columns 7 through 9 contain the contour shape parameters b (Half Width), S_z, and S_y. Finally, columns 10 and 11 contain the width from the centerline to the indicated concentration levels at the indicated height. Note that the output is prematurely terminated. Output actually continues until the centerline, ground-level concentration drops below the lowest concentration of interest.

```
     Gravity slumping velocity coefficient used: 1.150

     NON Isothermal calculation

     Heat transfer calculated with correlation:  1

     Water transfer not included


⑥               *****                 CALCULATED SOURCE PARAMETERS                  *****

      Time         Gas Radius      Height          Qstar        SZ(x=L/2.)   Mole frac C     Density      Temperature     Rich No.
      sec              m              m          kg/m**2/s         m                         kg/m**3           K

   0.000000E+00    22.0600     1.103030E-05  6.818148E-02  0.417234       1.00000       1.67985        112.008       0.756144
     1.60000       22.4197     1.511141E-02  6.761492E-02  0.424343      0.998022       1.66763        113.004       0.756144
     3.52000       23.1805     2.886016E-02  6.658905E-02  0.440051      0.994459       1.64163        115.114       0.756144

     4.48000       23.6237     3.352411E-02  6.604830E-02  0.448937      0.992791       1.62802        116.227       0.756144
     5.12000       23.9315     3.581163E-02  6.573497E-02  0.455514      0.991825       1.61802        117.033       0.756144
     6.40000       24.5615     3.841195E-02  6.514826E-02  0.468544      0.990368       1.59944        118.528       0.756144

     7.04000       24.8777     3.878484E-02  6.494597E-02  0.475763      0.989921       1.58919        119.334       0.756144
     8.96000       25.3717     3.822614E-02  6.489014E-02  0.489967      0.989604       1.56839        120.946       0.756144
    15.3600        25.1948     3.796661E-02  6.520404E-02  0.488793      0.989842       1.56810        120.946       0.756144


⑦  Source strength [kg/s] :                          130.00     Equivalent Primary source radius [m] :          22.060
   Equivalent Primary source length [m] :            44.120     Equivalent Primary source half-width [m] :      17.326

   Secondary source concentration [kg/m**3] :        1.5397     Secondary source SZ [m] :                       0.48879

   Contaminant flux rate:    5.11989E-02


   Secondary source mass fractions... contaminant:  0.981864       air:   1.80427E-02
           Enthalpy:  -3.86753E+05       Density:    1.5681

   Secondary source length [m] :                     50.390     Secondary source half-width [m] :               19.788


⑧  Distance     Mole   Concentration Density     Gamma    Temperature   Half       Sz          Sy     Width at z=  1.00 m to:
              Fraction                                                  Width                          5.00   mole%  15.0    mole%
     (m)                  (kg/m**3)  (kg/m**3)                (K)        (m)       (m)         (m)        (m)          (m)

     25.2      0.990        1.54        1.5681   0.322       121.       19.8      0.489    6.538E-06    19.8         19.8
     25.8      0.984        1.52        1.5523   0.316       123.       18.7      0.490       1.43      20.6         19.8
     29.2      0.962        1.44        1.5139   0.308       128.       17.9      0.490       3.75      22.7         20.5

     31.6      0.945        1.38        1.4982   0.309       131.       17.8      0.490       4.79      23.9         21.0
     34.8      0.923        1.30        1.4651   0.302       136.       17.9      0.495       5.93      25.3         21.7
     38.0      0.897        1.22        1.4261   0.289       143.       18.0      0.503       6.93      26.6         22.3
```

Notes on Transient Simulation Output of BURRO9

⑥ A summary of the calculated secondary source parameters is included. The secondary source gas radius and height are output as functions of time along with other secondary parameters including the source mass flux (Qstar), the vertical concentration distribution parameter at the downwind edge (SZ(x = L/2.)), the contaminant mole fraction (Mole frac C), the gas mixture density (Density), and the Richardson number based on the cloud spreading velocity (Rich No.). Note that only a portion of the output is shown. The source calculation ends when all the primary and secondary source gas has been taken up in the atmospheric flow.

⑥

```
            *****                 CALCULATED SOURCE PARAMETERS                    *****

   Time       Gas Radius      Height        Qstar      SZ(x=L/2.)   Mole frac C     Density    Temperature     Rich No.
   sec            m             m         kg/m**2/s        m                        kg/m**3          K

0.000000E+00   22.0600     1.103030E-05  6.818148E-02  0.417234       1.00000       1.67985      112.008      0.756144
1.250000E-03   22.0600     2.356897E-05  6.818144E-02  0.417234       1.00000       1.67985      112.008      0.756144
2.500000E-03   22.0600     3.610756E-05  6.818139E-02  0.417234       1.00000       1.67985      112.008      0.756144

3.750000E-03   22.0601     4.864604E-05  6.818133E-02  0.417234       1.00000       1.67985      112.008      0.756144
5.000000E-03   22.0601     6.118439E-05  6.818126E-02  0.417234       1.00000       1.67985      112.008      0.756144
6.250000E-03   22.0601     7.372261E-05  6.818118E-02  0.417234       0.999999      1.67985      112.008      0.756144

                                                   ●

                                                   ●

                                                   ●

  81.3600       14.8348    2.780102E-02  0.109961      0.606779       0.981068      1.24567      153.291      0.756144
  81.5200       12.0742    2.360015E-02  0.125116      0.589908       0.979136      1.19431      160.121      0.756144
  81.6800       8.90169    1.823564E-02  0.143168      0.529419       0.977271      1.13949      168.066      0.756144

  81.8400       5.27878    1.147265E-02  0.160261      0.386511       0.975869      1.07404      178.501      0.756144
  81.8600       4.74160    1.039801E-02  0.183404      0.399503       0.975860      1.06443      180.113      0.000000E+00
  81.9600       2.08103    4.857970E-03  0.172351      0.189777       0.975860      1.00028      191.664      0.000000E+00

  82.0100      0.828913    2.052904E-03  0.162497      8.258369E-02   0.975860      0.943090     203.287      0.000000E+00
  82.0325      0.296554    7.756192E-04  0.153897      3.261293E-02   0.975860      0.893179     214.645      0.000000E+00
  82.0337      0.267955    7.042261E-04  0.153154      2.975642E-02   0.975860      0.888863     215.686      0.000000E+00

  82.0350      0.245386    6.008303E-04  0.152315      2.748155E-02   0.975860      0.883995     216.873      0.000000E+00
  82.0378      0.245386    1.021048E-04  0.148447      2.748155E-02   0.975860      0.861546     222.516      0.000000E+00
  82.0383      0.245386    1.192880E-05  0.144329      2.748155E-02   0.975860      0.837649     228.849      0.000000E+00

  82.0384      0.245386    4.982423E-06  0.142887      2.748155E-02   0.975860      0.829278     231.150      0.000000E+00
```

⑦ An indication of the constants used in the x-direction dispersion correction is included. The output will also indicate if the x-direction dispersion correction was not applied.

⑧ The concentration field is shown for different times after the beginning of the spill. Unless otherwise specified, DEGADIS will choose default values for the time to output the concentration field. If other times are desired, DEGADIS3 can be executed again after the appropriate changes to RUN_NAME.ER3 have been made. See Appendix C for details.

⑨ The downwind portion of the calculations is included. The distance downwind of the source is given in the first column. Columns 2 through 6 contain the mole fraction, contaminant concentration, mixture density, ratio of $(\rho - \rho_a)/c_c$ (Gamma), and mixture temperature on the centerline of the gas cloud at ground level. Columns 7 through 9 contain the contour shape parameters b (Half Width), S_z, and S_y. Finally, columns 10 and 11 contain the width from the centerline to the indicated concentration levels at the indicated height. Note that the output is prematurely terminated.

```
    Sorted values for each specified time.

⑦   X-Direction correction was applied.
        Coefficient:        2.00000E-02
        Power:               1.2200
        Minimum Distance:    130.00     m


⑧    Time after beginning of spill   14.00000      sec

⑨ Distance    Mole   Concentration Density      Gamma   Temperature   Half        Sz          Sy     Width at z=  1.00 m to:
            Fraction                                                 Width                           5.00    mole%  15.0    mole%
    (m)                  (kg/m**3)  (kg/m**3)                   (K)     (m)        (m)         (m)       (m)          (m)

   36.4     0.929      1.06          1.2102   0.130        176.       16.0       0.594       6.25        24.3         20.9
   50.9     0.737      0.759         1.2468   0.230        187.       16.2       0.651       9.77        28.5         22.7
   65.7     0.521      0.456         1.2105   0.304        215.       16.4       0.647       12.7        29.6

   80.7     0.248      0.174         1.1286   0.324        263.       12.0       0.761       14.0        20.7

  For the UFL of     15.000     mole percent, and the LFL of     5.0000     mole percent:

  The mass of contaminant between the UFL and LFL is:   120.85     kg.
  The mass of contaminant above the LFL is:    786.92     kg.


     Time after beginning of spill   25.00000      sec

  Distance    Mole   Concentration Density      Gamma   Temperature   Half        Sz          Sy     Width at z=  1.00 m to:
            Fraction                                                 Width                           5.00    mole%  15.0    mole%
    (m)                  (kg/m**3)  (kg/m**3)                   (K)     (m)        (m)         (m)       (m)          (m)

   35.5     0.940      1.07          1.1932   0.113        178.       16.3       0.626       6.03        24.6         21.5
   50.0     0.788      0.826         1.2255   0.186        185.       16.4       0.701       9.66        29.4         24.2
   64.8     0.670      0.662         1.2472   0.265        192.       17.6       0.744       12.7        34.1         26.8

   79.8     0.592      0.544         1.2182   0.269        205.       19.0       0.775       15.3        38.1         28.6
   95.0     0.520      0.446         1.1846   0.253        220.       20.0       0.825       17.8        41.3         29.3
   110.     0.429      0.341         1.1567   0.249        236.       20.8       0.903       20.1        43.6         28.0
```

Preceding page blank

In addition to the output of the concentration field at specified times (using DEGADIS3), DEGADIS allows output of concentration time histories at specified positions using DEGADIS4 for transient releases. (Specifics about DEGADIS4 and its input requirements are included in Appendix C.) DEGADIS4 can be executed interactively or in batch mode. An example DEGADIS4 interactive run and output follow.

Notes on Using DEGADIS4

① DEGADIS4 can be run interactively as demonstrated here or in batch mode by supplying the same responses demonstrated here. DEGADIS4 sends the output to RUN_NAME.SR4.

② The RUN_NAME of the transient simulation is used here.

③ The number of downwind positions where output is desired is entered here. For each downwind distance, questions 4 and 5 are required.

④ The x-coordinate is the downwind distance from the source where output is desired.

⑤ At each downwind distance, DEGADIS4 asks for the y- and z-coordinates desired. DEGADIS4 automatically supplies information about the centerline concentration (y = 0 and z = 0). Note that negative numbers signal that no more positions are desired for this particular downwind distance.

```
①      $ RUN SYS$DEGADIS:DEGADIS4
②      Enter the file name used for this run: BURRO9
③      Enter the number of downwind distances desired:
        max of          10 downwind distances; 4 positions at each distance
       1

④      enter the x coordinate:
       100.
⑤        enter the y and z coordinate pairs at this distance:
       0., 1.
       30., 1.
       -1.,-1.
        t1:    17.000      tf:    98.562      dt:    2.0000      dist:     100.00
       FORTRAN STOP
       $
```

Notes in DEGADIS4 Output

① DEGADIS4 indicates the constants used in the x-direction dispersion correction. The user has the option of not making the x-direction dispersion correction if desired by making the appropriate changes to RUN_NAME.ER3. See Appendix C.

② At each downwind distance specified, the concentration profile parameters are output as a function of time. Note that part of the output has been removed.

```
①     X-Direction correction was applied.
          Coefficient:        2.00000E-02
          Power:               1.2200
          Minimum Distance:    130.00     m

②     Centerline values for the position -->
  x:   100.00     m


  Time      Mole    Concentration Density     Gamma   Temperature   Half        Sz          Sy     Width at z=  1.00 m to:
           Fraction                                                 Width                           5.00    mole%  15.0    mole%
   (s)                (kg/m**3)  (kg/m**3)                 (K)       (m)        (m)         (m)       (m)           (m)

  19.0     0.283      0.210      1.1320    0.300       259.        15.3      0.891        17.5       30.8
  21.0     0.385      0.302      1.1522    0.264       242.        18.5      0.861        18.3       37.7

  23.0     0.465      0.382      1.1713    0.260       229.        20.0      0.846        18.5       41.3        27.4
  25.0     0.491      0.411      1.1755    0.251       225.        20.2      0.850        18.5       42.2        29.3
  27.0     0.500      0.421      1.1759    0.246       224.        20.4      0.858        18.6       42.7        30.2

  29.0     0.502      0.423      1.1770    0.248       224.        20.6      0.868        18.7       43.2        30.7
  31.0     0.507      0.429      1.1780    0.246       223.        20.7      0.875        18.7       43.4        31.2
  33.0     0.511      0.435      1.1795    0.246       222.        20.6      0.877        18.7       43.4        31.4

  35.0     0.512      0.435      1.1791    0.245       222.        20.6      0.876        18.6       43.4        31.4
  37.0     0.511      0.433      1.1781    0.245       222.        20.6      0.876        18.7       43.4        31.3
  39.0     0.512      0.435      1.1794    0.246       222.        20.6      0.878        18.6       43.4        31.4

                                                        ●

                                                        ●

                                                        ●

  89.0     0.512      0.434      1.1779    0.243       222.        20.7      0.875        18.7       43.4        31.4
  91.0     0.511      0.434      1.1783    0.244       222.        20.6      0.876        18.7       43.4        31.4
  93.0     0.511      0.433      1.1778    0.245       222.        20.6      0.876        18.7       43.4        31.3

  95.0     0.485      0.401      1.1678    0.239       227.        20.9      0.893        19.2       44.0        31.1
```

③ DEGADIS4 outputs the concentration time history at the off-centerline positions specified.

③

```
    Time           Mole fraction at:        Mole fraction at:        Mole fraction at:        Mole fraction at:
                   y=  0.00000E+00 m        y=    30.000     m       y=   -1.0000     m       y=   0.00000E+00 m
    (s)            z=    1.0000    m        z=    1.0000     m       z=   -1.0000     m       z=   0.00000E+00 m
  17.00000           0.0000000E+00            0.0000000E+00
  19.00000           0.1060624                5.4025235E-02
  21.00000           0.1424979                9.8715379E-02

  23.00000           0.1730387                0.1328462
  25.00000           0.1860092                0.1447582
  27.00000           0.1921793                0.1513815

  29.00000           0.1952771                0.1557802
  31.00000           0.2000309                0.1600196
  33.00000           0.2026532                0.1617373

  35.00000           0.2026345                0.1614892
  37.00000           0.2019068                0.1608861
  39.00000           0.2030946                0.1616422

                                      ●

                                      ●

                                      ●

  89.00000           0.2021314                0.1614686
  91.00000           0.2021210                0.1613367
  93.00000           0.2015993                0.1608710

  95.00000           0.1926055                0.1577260
```

APPENDIX E

OOMS' MODEL CODE LISTING

Preceding page blank

```
C     *****PROGRAM EPA_NONISO_MAIN*****
C
C
      OPEN (UNIT=1,STATUS='OLD')
      OPEN (UNIT=3,STATUS='NEW')
C
      CALL STEADY
      CALL EXIT
      END
C
      INCLUDE 'sys$degadis:OOMS_STEADY.FOR/list'
      INCLUDE 'sys$degadis:OOMS.FOR/list'
C
C----------------------------------------------------------------------
      SUBROUTINE SETJET (YR,UR,QINIT,DIAJET,ROA,K1,K2,K3,ALOC,BLOC,DC)
C
C     THIS SUBROUTINE TRANSFORMS THE EXIT 'TOPHAT' VELOCITY PROFILE OF
C     A CONTINUOUS JET TO THE SIMILARITY (GAUSSIAN) FORM REQUIRED FOR
C     INPUT TO THE OOMS MODEL - USES WIND TUNNEL DATA CORRELATIONS BY
C     Y. KAMOTANI AND I. GREBER, NASA CONTRACTOR REPORT CR-2392, 3/74.
C
      REAL JAY
      DIMENSION YR(7),XOD(10),ZOD(10),PATHL(10)
      YR(4)=3.141592654/2.
      UV=1.2732*QINIT/((DIAJET/100.0)**2.0)
      JAY=(YR(1)/ROA)*(UV/UR)**2.0
      AV=EXP(0.405465+0.207519*ALOG(JAY))
      ROW=0.871667+0.1775*(UV/UR)
      XOD(0)=0.0
      ZOD(0)=0.0
      PATHL(0)=0.0
      I=0
  40  I=I+1
      XOD(I)=XOD(I-1)+0.10
      ZOD(I)=AV*XOD(I)**0.4
      PATHL(I)=PATHL(I-1)+SQRT((XOD(I)-XOD(I-1))**2.0+(ZOD(I)-
     1               ZOD(I-1))**2.0)
      IF (PATHL(I).LT.ROW) GO TO 40
      FXOD=XOD(I-1)+(XOD(I)-XOD(I-1))*((ROW-PATHL(I-1))/(PATHL(I)-
     1           PATHL(I-1)))
      YR(2)=(DIAJET/100.0)*SQRT(1.0/(4.0*(0.556796+0.486344*UR*
     1            COS(YR(4))/UV)))
      IF (K3.EQ.1.OR.K3.EQ.-1) GO TO 47
      YR(6)=FXOD*DIAJET/100.0
      IF (K1.NE.-1) GO TO 41
      YR(6)=-YR(6)
  41  IF (K2.EQ.1) GO TO 42
      IF (K2.EQ.-1) GO TO 43
      YR(7)=BLOC
      GO TO 44
  42  YR(7)=-(AV*((ABS(FXOD))**0.4))*(DIAJET/100.0)+BLOC
```

```
      GO TO 44
   43 YR(7)=(AV*((ABS(FXOD))**0.4))*(DIAJET/100.0)+BLOC
   44 CONTINUE
      IF (YR(7).LT.0.0) GO TO 45
      IF (YR(7).GT.DC) GO TO 45
      IF (K1.EQ.1 .OR. K1.EQ.-1) YR(4)=0.0
      IF (K2.EQ.1) YR(4)=ATAN(AV*0.4*(ABS(FXOD))**(-0.6))
      IF (K2.EQ.-1) YR(4)=-ATAN(AV*0.4*(ABS(FXOD))**(-0.6))
      YR(3)=UV-UR*COS(YR(4))
      GO TO 46
   45 YR(3)=UV-UR
      YR(4)=0.0
      YR(6)=0.0
      YR(7)=BLOC
      GO TO 46
   47 YR(6)=0.0
      YR(7)=BLOC
      ALOC=ALOC+FXOD*DIAJET/100.0
      IF (K3.EQ.1) YR(4)=ATAN(AV*0.4*(ABS(FXOD))**(-0.6))
      IF (K3.EQ.-1) YR(4)=-ATAN(AV*0.4*(ABS(FXOD))**(-0.6))
      YR(3)=UV-UR*COS(YR(4))
      YR(4)=0.0
   46 YR(7)=-YR(7)
      RETURN
      END
####
```

```
C     *****PROGRAM EPA_NONISO_STEADY*****
C
C
C     THIS PROGRAM CALCULATES THE TRAJECTORY AND DILUTION OF A
C     STEADY BUOYANT GAS JET RELEASED INTO A STATIONARY WIND FIELD
C
C     INPUT VARIABLES:
C
C   ACLASS(J) = PROBLEM DESCRIPTION
C        ALOC = DISTANCE IN THE Y DIRECTION FROM THE SECTOR CENTERLINE TO
C               THE JET ORIGIN, M
C        BLOC = VERTICAL DISTANCE FROM THE SECTOR SURFACE TO THE JET
C               ORIGIN, M
C          DC = DEPTH OF THE ATMOSPHERIC FLOW SECTOR, M
C        DENS = JET DENSITY, KG/M**3
C      DIAJET = INITIAL JET DIAMETER, CM
C       DISTA = INITIAL DOWNWIND DISTANCE TO THE FIRST PRINTOUT, M
C      DISTAN = INCREMENTAL DOWNWIND DISTANCE BETWEEN PRINTOUTS, M
C           H = INTEGRATION STEP SIZE, M
C          K1 = INITIAL JET DIRECTION INDICATOR(+1=HORIZONTAL DOWNWIND,
C               -1=HORIZONTAL UPWIND,0=ALL OTHER CONDITIONS)
C          K2 = INITIAL JET DIRECTION INDICATOR(+1=VERTICAL UPWARD,-1=
C               VERTICAL DOWNWARD,0=ALL OTHER CASES)
C          K3 = INITIAL JET DIRECTION INDICATOR(+1=HORIZONTAL TRANSVERSE
C               TO THE RIGHT,-1=HORIZONTAL TRANSVERSE TO THE LEFT,0=
C               ALL OTHER CASES)
C       QINIT = INITIAL JET FLOWRATE, M**3/S
C         ROA = AMBIENT AIR DENSITY, KG/M**3
C      TESTNO = TEST NUMBER
C          TA = AMBIENT TEMPERATURE, DEGREES CELSIUS
C          TG = INITIAL JET TEMPERATURE, DEGREES CELSIUS
C          UR = MEAN WIND VELOCITY, M/S
C           X = STARTING VALUE OF JET TRAJECTORY INTEGRATION, M
C
      SUBROUTINE STEADY
C
      INTEGER COUNT,RUNGE,HAMING
      REAL JAY,MWA,MWG,MOLEN,MW,MOLFR
      DOUBLE PRECISION ACLASS(4),TESTNO(1)
      DIMENSION TE(7),YR(7),FR(7),Y(4,7),F(3,7),YRS(7)
      DIMENSION XOD(10),ZOD(10),PATHL(10)
      COMMON/PHYS/DCDR,EPS,UPRI,ROA,DENS,G,UR,UA,CPA,CPG,MWA,MWG
      COMMON/COEFF/ALFA1,ALFA2,ALFA3
C
C     ..... READ IN DATA FROM FILE **.DAT - ASSIGN **.DAT FOR001 .....
      OPEN(UNIT=1,STATUS='OLD')
      READ(1,602) TESTNO(1)
      READ(1,603) (ACLASS(J),J=1,4)
      READ(1,600) X,H,DISTA,DISTAN
      READ(1,601) TA,TG,UR,ZO,ZJO
      READ(1,611) UST,ZR,MOLEN,ISTAB
      READ(1,601) MWA,MWG,CPA,CPG
      READ(1,604) DIAJET,QINIT
      READ(1,606) DC,ALOC,BLOC
      READ(1,609) K1,K2,K3
      READ(1,610) ALFA1,ALFA2,ALFA3
C
C     ..... INITIALIZE VARIABLES AND ASSIGN VALUES TO CONSTANTS .....
      G= 9.80665
      XINIT= X
      PI= 3.141592654
      P=101325.
      R=8314.
      ROA=P*MWA/R/TA
      DENS=P*MWG/R/TG
C
C     ..... PRINT VALUES OF PARAMETERS INITIAL CONDITIONS.....
      OPEN(UNIT=3,STATUS='NEW')
      WRITE(3,704) DISTA,DISTAN
      WRITE(3,736) ROA,DENS,TA,TG,DIAJET
      WRITE(3,705) QINIT,DC
      WRITE(3,706) ALOC,BLOC
      WRITE(3,712) UR
      IF (K1.EQ.1) WRITE(3,751)
      IF (K1.EQ.-1) WRITE(3,752)
      IF (K2.EQ.1) WRITE(3,753)
      IF (K2.EQ.-1) WRITE(3,754)
      IF (K3.EQ.1) WRITE(3,755)
      IF (K3.EQ.-1) WRITE(3,756)
      WRITE(3,719) ALFA1,ALFA2,ALFA3
      WRITE(3,726)
      WRITE(3,711)
C
C     ..... IF A VALUE FOR H, THE INTEGRATION STEPSIZE, HAS NOT .....
C     ..... INPUT, THEN SET H=0.01*DIAJET/5.0 .....
      IF (H.LE.0.001) H= DIAJET*0.002
      HH= H
C
C     ..... CALCULATE WIND SPEED AT THE JET DISCHARGE HEIGHT .....
      UA=UST/0.35*(ALOG((ZJ0+ZR)/ZR)-PSIF(ZJ0,MOLEN))
C
C     ..... INITIALIZE VARIABLES FOR OOMS MODEL .....
   13 DO 17 I=1,7
   17 YR(I)=0.0
      X= XINIT
      YR(1)=DENS
      DCDR=1./(1.-ROA/DENS)
      YR(5)=YR(1)/DCDR
C
C     ..... CALL SETJET TO CALCULATE THE INITIAL CONDITIONS .....
C     ..... FOR OOMS MODEL INTEGRATION .....
      CALL SETJET(YR,UA,QINIT,DIAJET,ROA,K1,K2,K3,ALOC,BLOC,DC)
```

```
C
      DO 20 I=1,7
      TE(I)=0.0
      Y(4,I)=YR(I)
      F(1,I)=0.0
      F(2,I)=0.0
      F(3,I)=0.0
   20 YRS(I)=0.0
      COUNT = 0
C
C     ..... CALCULATE CENTERLINE CONCENTRATION, WIDTH, VELOCITY .....
C     ..... DECREMENT, DENSITY DECREMENT, AND ANGLE OF TRAJECTORY .....
C     ..... Y(1) THROUGH Y(6) RESPECTIVELY .....
C
C     ..... CALL RUNGE TO INTEGRATE FOR COUNT.LE.3 .....
  100 IF (RUNGE(7,YR,FR,X,H).NE.1) GO TO 114
C
C     ..... CALCULATE WIND SPEED AT THE JET DISCHARGE HEIGHT .....
      UA=UST/0.35*(ALOG((ZJO+ZR)/ZR)-PSIF(ZJO,MOLEN))
C
C       ..... CALCULATE ATMOSPHERIC TURBULENCE ENTRAINMENT VELOCITY.....
      UPRI=UPRIME(ISTAB,UA,ZJO,Y(4,2))
C
C     ..... CALL RHS TO CALCULATE RIGHT HAND SIDES .....
      CALL RHS(YR,YRS)
      DO 101 K=1,4
  101 FR(K)=YRS(K)
      FR(5)=YRS(5)
      FR(6)=COS(YR(4))
      FR(7)=SIN(YR(4))
      GO TO 100
  114 COUNT = COUNT + 1
      ISUB = 4 - COUNT
      DO 117 J=1,7
  117 Y(ISUB,J) = YR(J)
C
C     ..... CALCULATE WIND SPEED AT THE POSITION OF THE JET .....
      ZJ=DC+YR(7)
      UA=UST/0.35*(ALOG((ZJ+ZR)/ZR)-PSIF(ZJ,MOLEN))
C
C       ..... CALCULATE ATMOSPHERIC TURBULENCE ENTRAINMENT VELOCITY.....
      UPRI=UPRIME(ISTAB,UA,ZJ,Y(ISUB,2))
C
      CALL RHS(YR,YRS)
      DO 120 K=1,4
  120 F(ISUB,K)=YRS(K)
      F(ISUB,5)=YRS(5)
      F(ISUB,6)= COS(YR(4))
      F(ISUB,7)= SIN(YR(4))
  123 CONTINUE
      IF (COUNT.LE.3) GO TO 133
      IF (ABS(Y(1,1)).LT.DENS/3.0) H=AMAX1(HH,H)
      IF ((DENS-ROA)*K2.GT.0.0.AND.Y(1,1).GT.DENS) GO TO 127
      IF ((DENS-ROA)*K1.GT.0.0.AND.Y(1,1).LT.0.0) GO TO 127
      GO TO 129
  127 K1=1
      WRITE(3,715)
      TYPE *,'JET DIRECTION CHANGED.'
      GO TO 13
  129 CONTINUE
C
C     ..... IF X.LE.DISTA, PRINT LOCATION AND PLUME CENTERLINE .....
C     ..... CONCENTRATION VALUES...RESET DISTA=DISTA+DISTAN .....
C     ..... IF THE JET REACHES THE GROUND, STOP THE COMPUTATION .....
C     ..... IF THE DW DISTANCE .GT.1000 M, STOP THE COMPUTATION .....
      IF (Y(1,6).GT.1000.)STOP
C
C     ..... CALCULATE THE PLUME CENTERLINE TEMPERATURE .....
      Q=MWA
      RATIO=Y(1,1)/(Y(1,5)+ROA)
      MOLFR=RATIO*Q/(RATIO*Q-MWG*(RATIO-1.))
      MW=MOLFR*MWG+(1.-MOLFR)*Q
      TEMPK=P*MW/R/(Y(1,5)+ROA)
C
      CTRHT=SQRT(2.)*Y(1,2)/COS(Y(1,4))
      IF ((Y(1,7)+DC-CTRHT).LE.0.1)WRITE(2,*) Y(1,6),Y(1,1),
     1                                      TEMPK,Y(1,2)
      IF ((Y(1,7)+DC-CTRHT).LE.0.1) STOP
      YCL=Y(1,1)
      IF (Y(1,6).GE.DISTA) WRITE(3,734) Y(1,6),(Y(1,7)+DC),YCL
      IF (Y(1,6).GE.DISTA) WRITE(6,9999) Y(1,6),-Y(1,7),YCL,
     1                            Y(1,2),Y(1,3),Y(1,4),Y(1,5)
      IF (Y(1,6).GE.DISTA) DISTA=DISTA+DISTAN
C
  133 CONTINUE
C
      IF (COUNT.LT.3) GO TO 100
C
C     ..... FOR COUNT.GT.3 CALL HAMING TO CONTINUE INTEGRATION .....
  137 M = HAMING(7,Y,F,X,H,TE)
      DO 140 K=1,5
  140 YR(K)=Y(1,K)
C
C     ..... CALCULATE WIND SPEED AT THE POSITION OF THE JET .....
      ZJ=DC+Y(1,7)
      UA=UST/0.35*(ALOG((ZJ+ZR)/ZR)-PSIF(ZJ,MOLEN))
C
C       ..... CALCULATE ATMOSPHERIC TURBULENCE ENTRAINMENT VELOCITY.....
      UPRI=UPRIME(ISTAB,UA,ZJ,Y(1,2))
C
C     ..... CALCULATE THE RIGHT HAND SIDES .....
      CALL RHS(YR,YRS)
      DO 143 K=1,5
  143 F(1,K)=YRS(K)
      F(1,6)=COS(Y(1,4))
      F(1,7)=SIN(Y(1,4))
      IF (M.EQ.1) GO TO 137
C
C     ..... INCREMENT STEP COUNTER AND CONTINUE INTEGRATION .....
      COUNT= COUNT+1
      IF (Y(1,7).GT.0.0) Y(1,7)=0.0
      IF (Y(1,7).LT.-DC) Y(1,7)=-DC
      GO TO 123
C
  200 CONTINUE
C
C     ..... FORMATS FOR INPUT STATEMENTS .....
   99 FORMAT(6X,I3)
  600 FORMAT(5X,F10.3,5X,F10.3,5X,F10.3,5X,F10.3)
  601 FORMAT(5X,F10.3,5X,F10.3,5X,F10.3,5X,F10.3,5X,F10.3)
  602 FORMAT(7X,A8)
  603 FORMAT(4A8)
  604 FORMAT(5X,F10.3,5X,E10.3,5X,F10.3)
  605 FORMAT(5X,F10.3,5X,F10.3)
  606 FORMAT(5X,F10.3,5X,F10.3,5X,F10.3)
  607 FORMAT(5X,F10.5,5X,F10.5,5X,F10.5)
  608 FORMAT(5X,F10.5,5X,F10.5,5X,F10.3,5X,F10.3)
  609 FORMAT(5X,I10,5X,I10,5X,I10)
  610 FORMAT(5X,F10.3,5X,F10.3,5X,F10.3,5X,F10.4)
  611 FORMAT(5X,F10.3,5X,E10.4,5X,E10.4,5X,I10)
C
C     ..... FORMATS FOR OUTPUT STATEMENTS .....
  704 FORMAT(/6X,40H0 DOWNWIND DISTANCE TO FIRST PRINTOUT = ,
     1       F4.1,2H M/6X,31H0 DISTANCE BETWEEN PRINTOUTS = ,F4.1,2H M)
  705 FORMAT(6X,25H0 INITIAL JET FLOWRATE = ,E11.4,7H M**3/S
     2       /6X,20H0 DEPTH OF SECTOR = ,F6.1,2H M)
  706 FORMAT(6X,48H0 DISTANCE FROM THE CENTERLINE TO THE JET ORIGIN
     1       /8X,21HIN THE Y DIRECTION = ,F6.1,2H M
     2       /6X,52H0 DISTANCE FROM THE SECTOR SURFACE TO THE JET ORIGIN
     3       /8X,21HIN THE Z DIRECTION = ,F6.1,2H M)
  708 FORMAT(/2X,10HTEST NO.: ,A8)
  709 FORMAT(/2X,13HDESCRIPTION: ,4A8/)
  711 FORMAT(/5X,16HJET DEVELOPMENT:
     1       //7X,1HX,14X,1HZ,10X,17HCONCENTRATION(CL)/4X,8H(METERS),7X,
     2       8H(METERS),10X,9H(KG/M**3)/)
  712 FORMAT(6X,30H0 MEAN VELOCITY OF THE WIND = ,F4.1,4H M/S/
     1       6X,24H0 ORIENTATION OF THE JET)
  715 FORMAT(/3X,21HJET DIRECTION CHANGED/)
  719 FORMAT(6X,26H0 ENTRAINMENT COEFFICIENTS/7X,11H    ALFA 1=,
     1       F8.4/7X,11H    ALFA 2=,F8.4/7X,11H    ALFA 3=,F8.4)
  726 FORMAT(/)
  734 FORMAT(2X,F12.4,3X,F12.3,7X,E11.4)
  736 FORMAT(6X,24H0 AMBIENT AIR DENSITY = ,F4.2,8H KG/M**3/6X,
     1      27H0 DISPERSION JET DENSITY = ,F4.2,8H KG/M**3/6X,
     2  24H0 AMBIENT TEMPERATURE = ,F5.1,12H DEG CELSIUS/6X,
     3      28H0 INITIAL JET TEMPERATURE = ,F5.1,12H DEG CELSIUS/6X,
     4      25H0 INITIAL JET DIAMETER = ,F5.1,3H CM)
  751 FORMAT(6X,24H     HORIZONTAL DOWNWIND)
  752 FORMAT(6X,22H     HORIZONTAL UPWIND)
  753 FORMAT(6X,20H     VERTICAL UPWARD)
  754 FORMAT(6X,22H     VERTICAL DOWNWARD)
  755 FORMAT(6X,39H     HORIZONTAL TRANSVERSE TO THE RIGHT)
  756 FORMAT(6X,38H     HORIZONTAL TRANSVERSE TO THE LEFT)
 9999 FORMAT (F10.2,F10.2,5F10.6)
      END
C
C-----------------------------------------------------------------
C     FUNCTION PSI
C
C*** PER COLENBRANDER ***
C
C*** THIS FUNCTION HAS BEEN DERIVED FROM BUSINGER,J.A.
C*** WORKSHOP ON MICROMETEOROLOGY, CHAPTER 2, HAUGEN,D.A. (ED.)
C*** AMERICAN METEOROLOGICAL SOCIETY.
C
      FUNCTION PSIF(Z,ML)
C
      include 'sys$degadis:DEGADIS1.dec'
c
      REAL*4 ML
C
      IF( ML ) 10,20,30
C
 10   A = (1.-15.*Z/ML)**.25
      PSIF = 2.*ALOG((1.+A)/2.) + ALOG((1.+A*A)/2.) - 2.*ATAN(A) +
     $ PI/2.
      RETURN
C
 20   PSIF = 0.
      RETURN
C
 30   PSIF = -4.7*Z/ML
      RETURN
      END
C
C-------------------------------------------------------------------
C     FUNCTION UPRIME
C
C*** ADAPTED FROM OOMS AND MAHIEU (1981):
C***   UPRIME = (EPSILON*B)**(1/3)
C***  EPSILON = 0.067*UA(Z)/Z (NEUTRAL) - BRIGGS (1969)
C***          = 0.004         (UNSTABLE) - KAIMAL ET AL. (1973)
C***          = 0.0           (STABLE) - KAIMAL ET AL. (1973)
C***        B = JET (RADIAL) CHARACTERISTIC LENGTH
```

```
C
      FUNCTION UPRIME(ISTAB,UA,Z,B)
C
      if(istab.le.0 .or. istab.gt.6) istab = 4
             if(istab.eq.1) then
              UPRIME=(0.004*B)**(1./3.)
             else if(istab.eq.2) then
              UPRIME=(0.004*B)**(1./3.)
             else if(istab.eq.3) then
              UPRIME=(0.004*B)**(1./3.)
             else if(istab.eq.4) then
              UPRIME=(0.067*UA/Z*B)**(1./3.)
             else if(istab.eq.5) then
              UPRIME=0.5*(0.067*UA/Z*B)**(1./3.)
             else if(istab.eq.6) then
              UPRIME=0.0
             endif
      RETURN
      END
####
```

```
C     *****PROGRAM EPA_NONISO_OOMS*****
C
C
      INTEGER FUNCTION HAMING( N,Y,F,X,H,TE)
C
C     FUNCTION HAMING IS TAKEN FROM 'APPLIED NUMERICAL METHODS' BY
C     B. CARNAHAN, H.A. LUTHER, AND J.O. WILKES, PUBLISHED BY J. WILEY
C     AND SONS, INC. 1969. PAGES 401 TO 402.
C
C     HAMING IMPLEMENTS HAMMING'S PREDICTOR-CORRECTOR ALGORITHM TO
C     SOLVE N SIMULTANEOUS FIRST-ORDER ORDINARY DIFFERENTIAL
C     EQUATIONS.  X IS THE INDEPENDENT VARIABLE AND H IS THE
C     INTEGRATION STEPSIZE.  THE ROUTINE MUST BE CALLED TWICE FOR
C     INTEGRATION ACROSS EACH STEP.  ON THE FIRST CALL, IT IS ASSUMED
C     THAT THE SOLUTION VALUES AND DERIVATIVE VALUES FOR THE N
C     EQUATIONS ARE STORED IN THE FIRST N COLUMNS OF THE FIRST
C     FOUR ROWS OF THE Y MATRIX AND THE FIRST THREE ROWS OF THE F
C     MATRIX, RESPECTIVELY.  THE ROUTINE COMPUTES THE N PREDICTED
C     SOLUTIONS YPRED(J), INCREMENTS X BY H AND PUSHES ALL
C     VALUES IN THE Y AND F MATRICES DOWN ONE ROW.  THE PREDICTED
C     SOLUTIONS YPRED(J) ARE MODIFIED, USING THE TRUNCATION ERROR
C     ESTIMATES TE(J) FROM THE PREVIOUS STEP, AND SAVED IN THE FIRST
C     ROW OF THE Y MATRIX.  HAMING RETURNS TO THE CALLING PROGRAM WITH
C     THE VALUE 1 TO INDICATE THAT ALL DERIVATIVES SHOULD BE COMPUTED
C     AND STORED IN THE FIRST ROW OF THE F ARRAY BEFORE THE SECOND
C     CALL IS MADE ON HAMING.  ON THE SECOND ENTRY TO THE FUNCTION
C     (DETERMINED BY THE LOGICAL VARIABLE PRED), HAMING USES THE
C     HAMMING CORRECTOR TO COMPUTE NEW SOLUTION ESTIMATES, ESTIMATES
C     THE TRUNCATION ERRORS TE(J) FOR THE CURRENT STEP, IMPROVES
C     THE CORRECTED SOLUTIONS USING THE NEW TRUNCATION ERROR
C     ESTIMATES, SAVES THE IMPROVED SOLUTIONS IN THE FIRST ROW OF THE
C     Y MATRIX, AND RETURNS TO THE CALLING PROGRAM WITH A VALUE 2 TO
C     INDICATE COMPLETION OF ONE FULL INTEGRATION STEP.
C
      LOGICAL PRED
      DIMENSION YPRED(20), TE(N), Y(4,N), F(3,N)
      DATA PRED /.TRUE./
C
C     ..... CALL FOR PREDICTOR OR CORRECTOR SECTION .....
      IF (.NOT.PRED) GO TO 4
C
C     ..... PREDICTOR SECTION OF HAMING .....
C     ..... COMPUTE PREDICTED Y(J) VALUES AT NEXT POINT .....
      DO 1 J=1,N
    1 YPRED(J) = Y(4,J) + 4.*H*(2.*F(1,J) - F(2,J) + 2.*F(3,J))/3.
C
C     ..... UPDATE THE Y AND F TABLES .....
      DO 2 J=1,N
      DO 2 K5=1,3
      K = 5 - K5
      Y(K,J) = Y(K-1,J)
    2 IF (K.LT.4) F(K,J) = F(K-1,J)
C
C     ..... MODIFY PREDICTED Y(J) VALUES USING THE TRUNCATION .....
C     ..... ERROR ESTIMATES FROM THE PREVIOUS STEP .....
C     ..... INCREMENT X VALUE .....
      DO 3 J=1,N
    3 Y(1,J) = YPRED(J) + 112.*TE(J)/9.
      X = X + H
C
C     ..... SET PRED AND REQUEST UPDATED DERIVATIVE VALUES .....
      PRED = .FALSE.
      HAMING = 1
      RETURN
C
C     ..... CORRECTOR SECTION OF HAMING .....
C     ..... COMPUTE CORRECTED AND IMPROVED VALUES OF THE Y(J) .....
C     ..... SAVE TRUNCATION ERROR ESTIMATES FOR CURRENT STEP .....
    4 DO 5 J=1,N
      Y(1,J) = (9.*Y(2,J)-Y(4,J) + 3.*H*(F(1,J)+2.*F(2,J)-F(3,J)))/8.
      TE(J) = 9.*(Y(1,J) - YPRED(J))/121.
    5 Y(1,J) = Y(1,J) - TE(J)
C
C     ..... SET PRED AND RETURN WITH SOLUTIONS FOR CURRENT STEP .....
      PRED = .TRUE.
      HAMING = 2
      RETURN
      END
C
C
      INTEGER FUNCTION RUNGE(N,Y,F,X,H)
C
C     FUNCTION RUNGE IS TAKEN FROM 'APPLIED NUMERICAL METHODS' BY
C     B. CARNAHAN, H.A. LUTHER, AND J.O. WILKES, PUBLISHED BY J. WILEY
C     AND SONS, INC. 1969. PAGES 374 TO 375.
C
C     THE FUNCTION RUNGE EMPLOYS THE FOURTH-ORDER RUNGE-KUTTA METHOD
C     WITH KUTTA'S COEFFICIENTS TO INTEGRATE A SYSTEM OF N SIMULTAN-
C     EOUS FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS F(J)=DY(J)/DX,
C     (J=1,2...,N), ACROSS ONE STEP OF LENGTH H IN THE INDEPENDENT
C     VARIABLE X, SUBJECT TO INITIAL CONDITIONS Y(J), (J=1,2,...,N).
C     EACH F(J), THE DERIVATIVE OF Y(J), MUST BE COMPUTED FOUR TIMES
C     PER INTEGRATION STEP BY THE CALLING PROGRAM.  THE FUNCTION MUST
C     BE CALLED FIVE TIMES PER STEP (PASS(1)...PASS(5)) SO THAT THE
C     INDEPENDENT VARIABLE VALUE (X) AND THE SOLUTION VALUES
C     (Y(1)...Y(N)) CAN BE UPDATED USING THE RUNGE-KUTTA ALGORITHM.
C     M IS THE PASS COUNTER.  RUNGE RETURNS AS ITS VALUE 1 TO
C     SIGNAL THAT ALL DERIVATIVES (THE F(J)) BE EVALUATED OR 0 TO
C     SIGNAL THAT THE INTEGRATION PROCESS FOR THE CURRENT STEP IS
C     FINISHED.  SAVEY(J) IS USED TO SAVE THE INITIAL VALUE OF Y(J)
C     AND PHI(J) IS THE INCREMENT FUNCTION FOR THE J(TH) EQUATION.
C     AS WRITTEN, N MAY BE NO LARGER THAN 50.
```

```
C
      DIMENSION PHI(50), SAVEY(50), Y(N), F(N)
      DATA M/0/
C
      M = M + 1
      GO TO (1,2,3,4,5), M
C
C     ..... PASS 1 .....
    1 RUNGE = 1
      RETURN
C
C     ..... PASS 2 .....
    2 DO 22 J=1,N
      SAVEY(J) = Y(J)
      PHI(J) = F(J)
   22 Y(J) = SAVEY(J) + 0.5*H*F(J)
      X = X + 0.5*H
      RUNGE = 1
      RETURN
C
C     ..... PASS 3 .....
    3 DO 33 J= 1,N
      PHI(J) = PHI(J) + 2.0*F(J)
   33 Y(J) = SAVEY(J) + 0.5*H*F(J)
      RUNGE = 1
      RETURN
C
C     ..... PASS 4 .....
    4 DO 44 J= 1,N
      PHI(J) = PHI(J) + 2.0*F(J)
   44 Y(J) = SAVEY(J) + H*F(J)
      X = X + 0.5*H
      RUNGE = 1
      RETURN
C
C     ..... PASS 5 .....
    5 DO 55 J = 1,N
   55 Y(J) = SAVEY(J) + (PHI(J) + F(J))*H/6.0
      M = 0
      RUNGE = 0
      RETURN
      END
C
C
      FUNCTION SIMUL(N,A,X,EPS,INDIC,NRC)
C
C     FUNCTION SIMUL  IS TAKEN FROM 'APPLIED NUMERICAL METHODS' BY
C     B. CARNAHAN, H.A. LUTHER, AND J.O. WILKES, PUBLISHED BY J. WILEY
C     AND SONS, INC. 1969. PAGES 290 TO 291.
C
C     WHEN INDIC IS NEGATIVE, SIMUL COMPUTES THE INVERSE OF THE N BY
```

```
C     N MATRIX A IN PLACE.  WHEN INDIC IS ZERO, SIMUL COMPUTES THE
C     N SOLUTIONS X(1)...X(N) CORRESPONDING TO THE SET OF LINEAR
C     EQUATIONS WITH AUGMENTED MATRIX OF COEFFICIENTS IN THE N BY
C     N+1 ARRAY A AND IN ADDITION COMPUTES THE INVERSE OF THE
C     COEFFICIENT MATRIX IN PLACE AS ABOVE.  IF INDIC IS POSITIVE,
C     THE SET OF LINEAR EQUATIONS IS SOLVED BUT THE INVERSE IS NOT
C     COMPUTED IN PLACE.  THE GAUSS-JORDAN COMPLETE ELIMINATION METHOD
C     IS EMPLOYED WITH THE MAXIMUM PIVOT STRATEGY.  ROW AND COLUMN
C     SUBSCRIPTS OF SUCCESSIVE PIVOT ELEMENTS ARE SAVED IN ORDER IN
C     THE IROW AND JCOL ARRAYS RESPECTIVELY.  K IS THE PIVOT COUNTER,
C     PIVOT THE ALGEBRAIC VALUE OF THE PIVOT ELEMENT, MAX
C     THE NUMBER OF COLUMNS IN A AND DETER THE DETERMINANT OF THE
C     COEFFICIENT MATRIX.  THE SOLUTIONS ARE COMPUTED IN THE (J+1) TH
C     COLUMN OF A AND THEN UNSCRAMBLED AND PUT IN PROPER ORDER IN
C     X(1)...X(N) USING THE PIVOT SUBSCRIPT INFORMATION AVAILABLE
C     IN THE IROW AND JCOL ARRAYS.  THE SIGN OF THE DETERMINANT IS
C     ADJUSTED, IF NECESSARY, BY DETERMINING IF AN ODD OR EVEN NUMBER
C     OF PAIRWISE INTERCHANGES IS REQUIRED TO PUT THE ELEMENTS OF THE
C     JORD ARRAY IN ASCENDING SEQUENCE WHERE JORD(IROW(I)) = JCOL(I).
C     IF THE INVERSE IS REQUIRED, IT IS UNSCRAMBLED IN PLACE USING
C     Y(1)...Y(N) AS TEMPORARY STORAGE.  THE VALUE OF THE DETERMINANT
C     IS RETURNED AS THE VALUE OF THE FUNCTION.  SHOULD THE POTENTIAL
C     PIVOT OR LARGEST MAGNITUDE BE SMALLER IN MAGNITUDE THAN EPS,
C     THE MATRIX IS CONSIDERED TO BE SINGULAR AND A TRUE ZERO IS
C     RETURNED AS THE VALUE OF THE FUNCTION.
C
      DIMENSION IROW(15),JCOL(15),JORD(15),Y(15)
      dimension A(6,6)      ! ts; workaround for access violation
      dimension X(5)              ! ts; workaround for access violation
c     dimension A(NRC,NRC)
c     dimension X(N)
C
      MAX=N
      IF(INDIC.GE.0) MAX=N+1
C
C     ..... BEGIN ELIMINATION PROCEDURE .....
      DETER=1
      DO 18 K=1,N
      KM1=K-1
C
C     ..... SEARCH FOR THE PIVOT ELEMENT .....
      PIVOT=0.
      DO 11 I=1,N
      DO 11 J=1,N
C
C     ..... SCAN IROW AND JCOL ARRAYS FOR INVALID PIVOT SUBSCRIPTS ...
      IF (K.EQ.1) GO TO 9
      DO 8 ISCAN=1,KM1
      DO 8 JSCAN=1,KM1
      IF (I.EQ.IROW(ISCAN)) GO TO 11
      IF (J.EQ.JCOL(JSCAN)) GO TO 11
```

```
    8 CONTINUE
    9 IF ( ABS(A(I,J)).LE. ABS(PIVOT)) GO TO 11
      PIVOT=A(I,J)
      IROW(K)=I
      JCOL(K)=J
   11 CONTINUE
C
C     ..... INSURE THAT SELECTED PIVOT IS LARGER THAN EPS .....
      IF ( ABS(PIVOT).GT.EPS) GO TO 13
      WRITE(3,202)
      SIMUL=0.
      RETURN
C
C     ..... UPDATE THE DETERMINANT VALUE .....
   13 IROWK=IROW(K)
      JCOLK=JCOL(K)
      DETER=DETER*PIVOT
C
C     ..... NORMALIZE PIVOT ROW ELEMENTS .....
      DO 14 J=1,MAX
   14 A(IROWK,J)=A(IROWK,J)/PIVOT
C
C     ..... CARRY OUT ELIMINATION AND DEVELOP INVERSE .....
      A(IROWK,JCOLK)=1./PIVOT
      DO 18 I=1,N
      AIJCK=A(I,JCOLK)
      IF (I.EQ.IROWK) GO TO 18
      A(I,JCOLK) = -AIJCK/PIVOT
      DO 17 J=1,MAX
   17 IF (J.NE.JCOLK) A(I,J)=A(I,J)-AIJCK*A(IROWK,J)
   18 CONTINUE
C
C     ..... ORDER SOLUTION VALUES (IF ANY) AND CREATE JORD ARRAY .....
      DO 20 I=1,N
      IROWI=IROW(I)
      JCOLI=JCOL(I)
      JORD(IROWI)=JCOLI
   20 IF (INDIC.GE.0) X(JCOLI)=A(IROWI,MAX)
C
C     ..... ADJUST SIGN OF DETERMINANT .....
      INTCH=0
      NM1=N-1
      DO 22 I=1,NM1
      IP1=I+1
      DO 22 J=IP1,N
      IF (JORD(J).GE.JORD(I)) GO TO 22
      JTEMP=JORD(J)
      JORD(J)=JORD(I)
      JORD(I)=JTEMP
      INTCH=INTCH+1
   22 CONTINUE
```

```
      IF (INTCH/2*2.NE.INTCH) DETER=-DETER

C     ..... IF INDIC IS POSITIVE RETURN WITH RESULTS .....
      IF (INDIC.LE.0) GO TO 26
      SIMUL=DETER
      RETURN
C
C     ..... IF INDIC IS NEGATIVE OR ZERO, UNSCRAMBLE THE INVERSE .....
C     ..... FIRST BY ROWS .....
   26 DO 28 J=1,N
      DO 27 I=1,N
      IROWI=IROW(I)
      JCOLI=JCOL(I)
   27 Y(JCOLI)=A(IROWI,J)
      DO 28 I=1,N
   28 A(I,J)=Y(I)

C     ..... THEN BY COLUMNS .....
      DO 30 I=1,N
      DO 29 J=1,N
      IROWJ=IROW(J)
      JCOLJ=JCOL(J)
   29 Y(IROWJ)=A(I,JCOLJ)
      DO 30 J=1,N
   30 A(I,J)=Y(J)
C
C     ..... RETURN FOR INDIC NEGATIVE OR ZERO .....
      SIMUL=DETER
      RETURN
  202 FORMAT(37H0SMALL PIVOT - MATRIX MAY BE SINGULAR)
      END
C
C
      SUBROUTINE RHS(Y,C)
      REAL MWA,MWG
      DIMENSION Y(7),A(6,6)
      dimension c(7)                 ! tos
      COMMON/PHYS/DCDR,EPS,UPRI,ROA,DENS,G,UR,UA,CPA,CPG,MWA,MWG
      COMMON/COEFF/ALFA1,ALFA2,ALFA3

C
C     ..... OOMS MODEL EQUATIONS .....
C
      ROA0=ROA
      DENSORC=DENS
      CSORC=DENS
      ST    =SIN(Y(4))
      ST2   =ST*ST
      CT    =COS(Y(4))
      CT2   =CT*CT
      UA2   =UA*UA
      UACT  =UA*CT
      UAST  =UA*ST
      Y12   =Y(1)*Y(2)
      Y22   =Y(2)*Y(2)
      Y33   =Y(3)*Y(3)
!     RHOSTR=Y(1)/DCDR
      RK1   =0.772699
      RK2   =0.412442
      RK3   =0.864665
      RK4   =1.043144
      RK5   =0.556796
      RK6   =1.729329
      RK7   =0.490842
      RK8   =1.113593
      RK9   =0.363346
      RK10  =0.981684
      RK11  =0.726692
      RK12  =3.458658
      RK13  =3.129432
      RK14  =2.227186
      RK15  =0.521572
      RK16  =0.432332
      RK17  =0.278398
C
C     ..... Gas Component Mass Balance Equation .....
C
      A(1,1)=Y(2)*(RK1*UACT+RK2*Y(3))
      A(1,2)=2.*Y(1)*(RK1*UACT+RK2*Y(3))
      A(1,3)=RK2*Y12
      A(1,4)=-RK1*UA*Y12*ST
      A(1,5)=0.
      A(1,6)=0.
C
C     ..... Overall Mass Balance Equation .....
C
        A(2,1)=0.
      A(2,2)=2.*(2.*UACT+RK3*Y(3)+RK4*Y(5)*UACT/ROA
     2         +RK5*Y(5)*Y(3)/ROA)
      A(2,3)=Y(2)*(RK3+RK5*Y(5)/ROA)
      A(2,4)=Y(2)*(-2.*UAST-RK4*Y(5)*UAST/ROA)
        A(2,5)=Y(2)*(RK4*UACT+RK5*Y(3))/ROA
      A(2,6)=2.*(ALFA1*ABS(Y(3))+ALFA2*UACT*ABS(ST)+ALFA3*UPRI)
C
C     ..... X-Momentum Balance Equation .....
C
      A(3,1)=0.
      A(3,2)=2.*Y(3)*CT*(RK6*UACT+RK7*Y(3)+Y(5)/ROA*(RK8*UACT
     2         +RK9*Y(3)))+UACT*UACT*CT*(4.+2.*RK4*Y(5)/ROA)
      A(3,3)=Y(2)*CT*(RK6*UACT+RK10*Y(3)
     2         +Y(5)/ROA*(RK8*UACT+RK11*Y(3)))
      A(3,4)=Y(2)*UACT*ST*(-6.*UACT-RK12*Y(3)-RK13*UACT*Y(5)/ROA
     2          -RK14*Y(3)*Y(5)/ROA)-Y(2)*Y33*ST*(RK7+RK9*Y(5)/ROA)
        A(3,5)=Y(2)*(RK4*UACT*UACT*CT+RK8*UACT*Y(3)*CT+RK9*Y33*CT)/ROA
      A(3,6)=UA*A(2,6)+0.3*UA2*ABS(ST*ST*ST)
C
C     ..... Y-Momentum Balance Equation .....
C
      A(4,1)=0.
      A(4,2)=2.*CT*ST*(2.*UA*UACT+RK6*UA*Y(3)
     2          +RK4*UA*UACT*Y(5)/ROA+RK8*UA*Y(3)*Y(5)/ROA)
     3          +2.*ST*(RK7*Y33+RK9*Y33*Y(5)/ROA)
      A(4,3)=Y(2)*ST*(RK6*UACT+RK10*Y(3)
     2          +Y(5)/ROA*(RK8*UACT+RK11*Y(3)))
      A(4,4)=Y(2)*UA2*CT*(1.-3.*ST2)*(2.+RK4*Y(5)/ROA)
     2          +Y(2)*RK7*Y33*CT+Y(2)*(CT2-ST2)*UA*Y(3)
     3          *(RK6+RK8*Y(5)/ROA)+Y(2)*RK9*Y33*CT*Y(5)/ROA
      A(4,5)=Y(2)*(RK4*UACT*UACT*ST+RK8*Y(3)*ST*CT*UA
     1          +RK9*Y33*ST)/ROA
      IF (ST.LT.0.) GO TO 40
      SIG=-1.
      GO TO 45
   40 SIG=1.
   45 CONTINUE
      A(4,6)=-RK4*Y(2)*G*Y(5)/ROA + SIG*0.3*UAST*UAST*CT
C
C     ..... Energy Balance Equation .....
C
      GROUP1=MWA/MWG-1.
      GROUP2=MWG/MWA*CPG/CPA
      GROUP3=1-ROA/ROA0
      GROUP4=DENSORC/CSORC/ROA0
      RTSIDE=0.
C
      A(5,1)=-RK15*GROUP1*GROUP4*UA*Y22*CT
     1          -RK17*GROUP1*GROUP4*Y22*Y(3)
      A(5,2)=2.*Y(2)*(UA*GROUP3*CT-RK15*UA/ROA0*CT*Y(5)
     1          -RK15*GROUP1*GROUP4*UA*Y(1)*CT+RK16*GROUP3*Y(3)
     2              -RK17/ROA0*Y(5)*Y(3)-RK17*GROUP1*GROUP4*Y(3)*Y(1))
      A(5,3)=Y22*(RK16*GROUP3-RK17/ROA0*Y(5)
     1          -RK17*GROUP1*GROUP4*Y(1))
      A(5,4)=-ST*(GROUP3*UA*Y22-RK15*UA/ROA0*Y22*Y(5)
     1          -RK15*GROUP1*GROUP4*UA*Y22*Y(1))
      A(5,5)=-RK15*UA/ROA0*Y22*CT-RK17/ROA0*Y22*Y(3)
      A(5,6)=RTSIDE/((GROUP2+1.)*3.141592654)
C
C        SIMUL USED FOR MATRIX INVERSION
C
      DETER=SIMUL(5,A,C,EPS,1,6)
      RETURN
      END
$$$$
```

APPENDIX F

DEGADIS CODE LISTING

AFGEN.FOR F-3
AFGEN2.FOR F-4
ALPH.FOR F-5
CRFG.FOR F-8
DEGADIS1.DEC F-13
DEGADIS1.FOR F-14
DEGADIS2.DEC F-22
DEGADIS2.FOR F-23
DEGADIS3.DEC F-27
DEGADIS3.FOR F-28
DEGADIS4.DEC F-32
DEGADIS4.FOR F-33
DEGADISIN.DEC F-38
DEGADISIN.FOR F-39
DOSOUT.FOR F-44
ESTRT1.FOR F-49
ESTRT2.FOR F-53
ESTRT2SS.FOR F-55
ESTRT3.FOR F-57
GAMMA.FOR F-58
GETTIM.FOR F-60
GETTIMDOS.FOR F-62
HEAD.FOR F-63
INCGAMMA.FOR F-69
IO.FOR F-73
IOT.FOR F-75
NOBL.FOR F-92
OB.FOR F-95
PSIF.FOR F-98
PSS.FOR F-99
PSSOUT.FOR F-102
PSSOUTSS.FOR F-105
RIPHIF.FOR F-108
RKGST.FOR F-111
RTMI.FOR F-119
SDEGADIS2.FOR F-123
SERIES.FOR F-131
SORTS.FOR F-132
SORTS1.FOR F-135
SRC1.FOR F-140
SRTOUT.FOR F-149
SSG.FOR F-153
SSGOUT.FOR F-156
SSGOUTSS.FOR F-159
SSOUT.FOR F-162
SSSUP.FOR F-164
STRT2.FOR F-173
STRT2SS.FOR F-176
STRT3.FOR F-178
SURFACE.FOR F-180
SZF.FOR F-182
TPROP.FOR F-184
TRANS1.FOR F-199
TRANS2.FOR F-202
TRANS2SS.FOR F-204
TRANS3.FOR F-206
TRAP.FOR F-207
TS.FOR F-216
TUPF.FOR F-217
UIT.FOR F-222

Preceding page blank

```
C............................................................
C
C     THIS FUNCTION LINEARLY INTERPOLATES FROM THE GIVEN
C         PAIR OF DATA POINTS.
C
      FUNCTION AFGEN(TAB,X,SPEC)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS1.dec'
      common/nend/poundn,pound
c
      character*4 pound
      character*(*) SPEC
      DIMENSION TAB(1)
C
      IF(X .GE. TAB(1)) GO TO 95
      WRITE(lunlog,1100) x,spec
      AFGEN = TAB(2)
      RETURN
C
 95   continue
      ix = 1
  100 Ix = ix + 2
C
      IY = IX + 1
      IF( TAB(IX).EQ.POUNDN .AND. TAB(IY).EQ.POUNDN ) GO TO 500
      IF(X .GE. TAB(IX)) GO TO 100
C
      IXP = IX-2
      IYP = IXP + 1
C
      SL = (TAB(IY) - TAB(IYP))/(TAB(IX) - TAB(IXP))
      AFGEN = SL*(X - TAB(IXP)) + TAB(IYP)
      RETURN
C
  500 CONTINUE
      IX = IX-2
      IY = IY-2
      IXP = IX-2
      IYP = IY-2
C
      SL = (TAB(IY) - TAB(IYP))/(TAB(IX) - TAB(IXP))
      AFGEN = SL*(X - TAB(IXP)) + TAB(IYP)
C
 1100 FORMAT(2X,'?AFGEN? UNDERFLOW; argument: ',1pg13.5,5X,A20)
      RETURN
      END
####
```

```
C..............................................................
C
C     THIS FUNCTION LINEARLY INTERPOLATES FROM THE GIVEN
C         PAIR OF DATA POINTS.
C
      FUNCTION AFGEN2(XTAB,TAB,X,SPEC)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )


C
      include 'sys$degadis:DEGADIS1.dec'
      common/nend/poundn,pound
c
      character*4 pound
      character*(*) SPEC
      DIMENSION XTAB(1),TAB(1)
C
      IF(X .GE. XTAB(1)) GO TO 95
      WRITE(lunlog,1100) x,spec
      AFGEN2 = TAB(2)
      RETURN
C
 95   continue
      ix = 1
  100 Ix = ix + 1
C
      IF( XTAB(IX).EQ.POUNDN ) GO TO 500
      IF(X .GE. XTAB(IX)) GO TO 100
C
      IXP = IX-1
C
      SL = (TAB(IX) - TAB(IXP))/(XTAB(IX) - XTAB(IXP))
      AFGEN2 = SL*(X - XTAB(IXP)) + TAB(IXP)
      RETURN
C
  500 CONTINUE
      IX = IX-1
      IXP = IX-1
C
      SL = (TAB(IX) - TAB(IXP))/(XTAB(IX) - XTAB(IXP))
      AFGEN2 = SL*(X - XTAB(IXP)) + TAB(IXP)
C
 1100 FORMAT(2X,'?AFGEN2? UNDERFLOW; argument: ',1pg13.5,5X,A20)
      RETURN
      END
####
```

```
C...............................................................
C
C     SUBROUTINES TO CALCULATE THE VALUE OF ALPHA
C
      SUBROUTINE ALPH

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )


C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ALP/ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
C
      REAL*8 ML,K
C
      EXTERNAL ALPHI
C
      PSI = PSIF(Z0,ML)
      USTAR = U0*K/(dLOG((Z0+ZR)/ZR) - PSI)
c
      if(u0 .eq. 0.) then
            alpha = 0.
            ustar = 0.
            return
            endif
c
      if(ialpfl.eq. 0) then
            alpha = alpco
            return
            endif
C
C*** RMTI USED TO DETERMINE THE ROOT OF THE REQUIRED INTEGRAL EQUATION
C
      IEND = 40
      IER = 0
C
      CALL RTMI(X,F,ALPHI,XLI,XRI,EPS,IEND,IER)
      IF( IER .NE. 0 ) CALL trap(19,IER)
      ALPHA = X
C
      RETURN
      END
c
c
C...................................................................
```

```
C
C     FUNCTION TO EVALUATE THE WEIGHTED EUCLIDEAN NORM OF THE
C      ERROR ASSOCIATED WITH THE POWER LAW FIT OF THE WIND PROFILE.
C
      FUNCTION ALPHI(X)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ALP/ALPHA,alpha1
C
      REAL*8 ML,K
C
      DIMENSION Y(1),DERY(1),PRMT(5),AUX(8)
      EXTERNAL ARG,ARGOUT
C
      ALPHA = X
C
      PRMT(1) = Z0
      PRMT(2) = dmax1(ZLOW,zr)    ! to take care of large zr
      PRMT(3) = STPINZ
      PRMT(4) = ERBNDZ
      PRMT(5) = STPMXZ
C
      Y(1) = 0.0D00
C
      DERY(1) = 1.0D00
C
      NDIM = 1
      IHLF = 0
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,ARG,ARGOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(18,IHLF)
      ALPHI = Y(1)
      RETURN
      END
C
C
C.............................................................
C
C     FUNCTION TO EVALUATE THE ARGUMENT OF THE INTEGRAL EXPRESSION
C
      SUBROUTINE ARG(Z,Y,D,PRMT)
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ALP/ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
C
      REAL*8 ML,K
C
      DIMENSION Y(1),D(1),PRMT(1)
C
C*** WEIGHT FUNCTION USED
C
      W = 1.D00/(1.D00 + Z)
      if(ialpfl.eq. 2) w= 1.D00
C
C*** WIND VELOCITY @ Z -- BEST FIT
C
      UBST = USTAR/K*(dLOG((Z+ZR)/ZR) - PSIF(Z,ML))
C
C*** WIND VELOCITY @ Z -- POWER LAW APPROXIMATION
C
      UALP = U0 * (Z/Z0) ** ALPHA
C
      D(1) = W * (UBST - UALP) * dLOG(Z/Z0) * UALP
      RETURN
      END
c
C
      SUBROUTINE ARGOUT
      RETURN
      END
$$$$
```

```
C...............................................................
C
C     SUBROUTINE TO CREATE RADG,QSTR,srcden,srcwc,srcwa,srcenth DATA LISTS
C
C     PARAMETERS  --       TABLE - WORKSPACE VECTOR
C           NTAB - DIMENSION OF TABLE DIVIDED BY iout_src
C           RER - RELATIVE ERROR BOUND OF CREATED
C                 DATA PAIRS BY LINEAR INTERPOLATION
C
C     VALUES OF TIME, RADG, height, QSTR, SZ0, yc, ya, rho, Ri,
c          wc,wa,enthalpy,temp
c       ARE READ INTO
C       TABLE(1) TO TABLE(13) RESPECTIVELY.
C
C...............................................................
C
      SUBROUTINE CRFG(TABLE,NTAB,rer)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      DIMENSION TABLE(1)
C
      include 'sys$degadis:DEGADIS1.dec'
      parameter (zero= 1.e-20)
c
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/NEND/ POUNDN,POUND
c
      character*4 pound
c
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      DATA NI/1/
c
      data iti/1/   ! time - element no 1 in record
      data irg/2/   ! Radg - element no 2 in record
      data iqs/4/   ! Qstar - element no 4 in record
      data idn/8/   ! rho - element no 8 in record
      data iwc/10/  ! wc - element no 10 in record
      data iwa/11/  ! wa - element no 11 in record
      data ien/12/  ! enthalpy - element no 12 in record
c
C
C     OUTPUT CREATED VECTORS TO A PRINT FILE
```

```
C
C
      READ(9,*) (TABLE(J),J=1,iout_src)
C
      WRITE(8,1111)
      WRITE(8,1105)
      if(isofl.eq. 1) then
      WRITE(8,1102)
      WRITE(8,1103)
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(9)
      else
      WRITE(8,1100)
      WRITE(8,1104)
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(13),table(9)
      endif
      ispace = 1
C
 1100   FORMAT(/,5X,'Time',5X,2x,'Gas Radius',2x,4X,'Height',4X,
     $4x,'Qstar',5x,2x,'SZ(x=L/2.)',2x,1X,'Mole frac C',2x,
     $3x,'Density',4x,1x,'Temperature',2x,3x,'Rich No.',3x)
 1102   FORMAT(/,5X,'Time',5X,2x,'Gas Radius',2x,4X,'Height',4X,
     $4x,'Qstar',5x,2x,'SZ(x=L/2.)',2x,1X,'Mole frac C',2x,
     $3x,'Density',4x,3x,'Rich No.',3x)
 1103 FORMAT(1H ,4X,'sec',6X,6X,'m',7X,6X,'m',7X,
     $2X,'kg/m**2/s',3X,6X,'m',7X,14x,3x,'kg/m**3',4x,14x,/)
 1104 FORMAT(1H ,4X,'sec',6X,6X,'m',7X,6X,'m',7X,
     $2X,'kg/m**2/s',3X,6X,'m',7X,14x,3x,'kg/m**3',4x,6x,'K',7x,14x,/)
 1105 FORMAT(1H ,23X,'*****',21X,'CALCULATED SOURCE PARAMETERS',21X,
     $'*****')
C
      RADG(1,1) = 0.
      RADG(2,1) = TABLE(2)
      QSTR(1,1) = 0.
      QSTR(2,1) = TABLE(4)
      srcden(1,1) = 0.
      srcden(2,1) = table(8)
      srcwc(1,1) = 0.
      srcwc(2,1) = table(10)
      srcwa(1,1) = 0.
      srcwa(2,1) = table(11)
      srcenth(1,1) = 0.
      srcenth(2,1) = table(12)
C
      READ(9,*) (TABLE(J),J=1,iout_src)
      L = 2
C
C*** L IS THE NUMBER OF RECORDS WHICH HAVE BEEN READ
C
  100   CONTINUE
      DO 120 I=2,NTAB
C
```

```
C*** MOVE LAST RECORD READ INTO THE LAST ACTIVE POSITION OF TABLE
C
      DO 130 J = 1,iout_src
      KK = iout_src * (I-1) + J
  130   TABLE(KK) = TABLE(J)
      KK = iout_src * I
C
C*** READ THE NEXT RECORD.  INCREMENT L.
C
      L = L + 1
      READ(9,*,END=900) (TABLE(J),J=1,iout_src)
C
C
      DO 140 Kkk = 2,I
C
      KT    = iout_src*(Kkk-1) + iti
      KRG   = iout_src*(Kkk-1) + irs
      KQSTR = iout_src*(Kkk-1) + iqs
      KCA   = iout_src*(Kkk-1) + idn
      Kwc   = iout_src*(Kkk-1) + iwc
      Kwa   = iout_src*(Kkk-1) + iwa
      Ken   = iout_src*(Kkk-1) + ien
C
      timeslot = rads(1,ni)
      ratio = (table(kt) - timeslot) / (table(iti) - timeslot)
C
      ANSRG = (TABLE(irs) - RADG(2,NI)) * ratio + RADG(2,NI)
      ANSQ = (TABLE(iqs) - QSTR(2,NI)) * ratio + QSTR(2,NI)
      ANSCA = (TABLE(idn) - srcden(2,NI)) * ratio + srcden(2,NI)
      ANSwC = (TABLE(iwc) - srcwc(2,NI)) * ratio + srcwc(2,NI)
      ANSwa = (TABLE(iwa) - srcwa(2,NI)) * ratio + srcwa(2,NI)
      ANSen = (TABLE(ien) - srcenth(2,NI)) * ratio + srcenth(2,NI)
C
      ERRG = ABS(ANSRG - TABLE(KRG))/TABLE(KRG)
      ERQSTR = ABS(ANSQ - TABLE(KQSTR))/(TABLE(KQSTR)+zero)
      ERO = dMAX1(ERRG,ERQSTR)
C
      ERCA = ABS(ANSCA - TABLE(KCA))/TABLE(KCA)
      ERO = dMAX1(ERO,ERCA)
C
      ERwC = ABS(ANSwC - TABLE(KwC))/(TABLE(KwC)+ zero)
      ERO = dMAX1(ERO,ERwC)
      ERwa = ABS(ANSwa - TABLE(Kwa))/(TABLE(Kwa)+ zero)
      ERO = dMAX1(ERO,ERwa)
      ERen = ABS(ANSen - TABLE(Ken))/(TABLE(Ken)+ zero)
      ERO = dMAX1(ERO,ERen)
C
      IF(ERO .GT. RER) GO TO 150
  140   CONTINUE
  120   CONTINUE
C
```

```
      WRITE(lunlog,1110)
C
C*** RECORD NEXT DATA PAIR. SINCE ERROR EXCEEDED, RECORD THE LAST
C***  DATA PAIR WHICH SATISFIED THE ERROR CRITERIA WHICH IS STORED
C***  IN TABLE(KK-(iout_src-1)) TO TABLE(KK)
C
  150   NI = NI + 1
c
      KT    = KK - iout_src + iti
      KRG   = KK - iout_src + irs
      KQSTR = KK - iout_src + ias
      KCA   = KK - iout_src + idn
      KwC   = KK - iout_src + iwc
      Kwa   = KK - iout_src + iwa
      Ken   = KK - iout_src + ien
c
      IF(NI .GT. MAXL) then
            write(lunlog,*) ' CRFG? Time out! ',table(kt)
            CALL trap(5)
            endif
C
      RADG(1,NI)   = TABLE(KT)
      RADG(2,NI)   = TABLE(KRG)
      QSTR(1,NI)   = TABLE(KT)
      QSTR(2,NI)   = TABLE(KQSTR)
      srcden(1,NI) = TABLE(KT)
      srcden(2,NI) = TABLE(KCA)
      srcwc(1,NI)  = TABLE(KT)
      srcwc(2,NI)  = TABLE(KWC)
      srcwa(1,NI)  = TABLE(KT)
      srcwa(2,NI)  = TABLE(KWA)
      srcenth(1,NI)  = TABLE(KT)
      srcenth(2,NI)  = TABLE(KEN)
C
C*** WRITE THE POINTS JUST RECORDED TO UNIT=8
C
      if(isofl.eq. 1) then
      WRITE(8,1140) (TABLE(J),J=kt,kt+5),table(kt+7),table(kt+8)
      else
      WRITE(8,1140) (TABLE(J),J=kt,kt+5),table(kt+7),
      1     table(kt+12),table(kt+8)
      endif
      ispace = ispace + 1
      if(ispace.eq. 3) then
            ispace = 0
            write(8,1111)
            endif
C
      GO TO 100
C
  900 CONTINUE             ! EOF encountered
```

```
C
      NI   = NI   + 1
      IF(NI+1 .GT. MAXL) then
            write(lunlog,*) ' CRFG? Time out: ',table(iti)
            CALL trap(5)
            endif
c
      RADG(1,NI)   = TABLE(iti)
      RADG(2,NI)   = TABLE(irg)
      QSTR(1,NI)   = TABLE(iti)
      QSTR(2,NI)   = TABLE(iqs)
      srcden(1,NI) = TABLE(iti)
      srcden(2,NI) = TABLE(idn)
      srcwc(1,NI)  = TABLE(iti)
      srcwc(2,NI)  = TABLE(iwc)
      srcwa(1,NI)  = TABLE(iti)
      srcwa(2,NI)  = TABLE(iwa)
      srcenth(1,NI)  = TABLE(iti)
      srcenth(2,NI)  = TABLE(ien)
c
      if(isofl.eq. 1) then
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(9)
      else
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(13),table(9)
      endif
      ispace = ispace + 1
      if(ispace.eq. 3) then
            ispace = 0
            write(8,1111)
            endif
c
      NI   = NI   + 1
      DO 910 I =1,2
      RADG(I,NI)    = POUNDN
      QSTR(I,NI)    = POUNDN
      srcden(I,NI)  = POUNDN
      srcwc(I,NI)   = POUNDN
      srcwa(I,NI)   = POUNDN
      srcenth(I,NI) = POUNDN
  910 CONTINUE
C
      RETURN
C
 1110 FORMAT(' ?CRFG? TABLE exceeded without point selection ',
     $'- execution continuing')
 1111 FORMAT(1H )
c1140 FORMAT(1H ,<iout_src>(1PG13.6,1X))
 1140 FORMAT(1H ,9(1PG13.6,1X))
      END
####
```

```
C....................................................
C
C
C     DIMENSIONS/DECLARATIONS for DEGADIS1
C
      include 'sys$degadis:DEGADISIN.dec'
c
c  maxl is the length of the /GEN3/ output vectors
c
      parameter (   lunlog= 6,
     $              sqrtpi= 1.77 245 3851D0,        ! sqrt(pi)
     $              maxl= 800,
     $              maxl2= 2*maxl,
     $              iout_src= 13)
C
####
```

```
      PROGRAM DEGADIS1
C
C***************************************************************************
C***************************************************************************
C***************************************************************************
C
C     Program description:
C
C     DEGADIS1 estimates the ambient wind profile power alpha and
C     characterizes the primary gas source.
C
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled 'Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
C
C     Disclaimer:
C
C     This computer code material was prepared by the University of
C     Arkansas as an account of work sponsored by the U. S. Coast Guard
C     and the Gas Research Institute.  Neither the University of Arkansas,
C     nor any person acting on its behalf:
C
C     a.  Makes any warranty or representation, express or implied,
C         with respect to the accuracy, completeness, or usefulness
C         of the information contained in this computer code material,
C         or that the use of any apparatus, method, numerical model,
C         or process disclosed in this computer code material may not
C         infringe privately owned rights; or
C
C     b.  Assumes any liability with respect to the use of, or for
C         damages resulting from the use of, any information,
C         apparatus, method, or process disclosed in this computer
C         code material.
C
```

```
C
C**********************************************************************
C**********************************************************************
C**********************************************************************
C
C
C     DIMENSIONS/DECLARATIONS
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'
C
c  ntab is the dimension of table divided by iout_src
c
      parameter (   ntab0=910,
     $              ntab=ntab0/iout_src)
c
      include '($ssdef)'
C
C     BLOCK COMMON
C
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/SZFC/ szstp0,szerr,szstpmx,szsz0
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/vucom/ vua,vub,vuc,vud,vudelta,vuflag
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/com_ENTHAL/ H_masrte,H_airrte,H_watrte
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUTS
```

```
C
C
      character*80 TITLE(4)
C
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
      character*3 gas_name
C
      real*4 tt1
      REAL*8 ML,K
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      logical vuflag
      logical reflag
C
      REAL*8 L,L0
      DIMENSION PRMT(25),Y(7),DERY(7),AUX(8,7)
      EXTERNAL SRC1,SRC10
      character*40 opnrup1
      character OPNRUP(40)
      character*4 INP,ER1,SCD,TR2,scl
c
      dimension table(ntab0)
c
      equivalence(opnrup(1),opnrup1)
c
C.....................................................
C
C     DATA
C
      DATA POUNDN/-1.D-20/,POUND/'//  '/
      DATA USTAR/0.D0/,GAMMAF/0.D0/
      DATA G/9.81D0/, K/0.35D0/
      DATA GAMMAF/0.D0/
      DATA PRMT/25*0.D0/
      DATA Y/7*0.D0/, DERY/7*0.D0/
      DATA TIME0/0.D0/,NDIM/0/
      DATA EMAX/0.D0/,TSC1/0.D0/
      DATA PTIME/igen*0.D0/
      DATA ET/igen*0.D0/,R1T/igen*0.D0/
      DATA PWC/igen*0.D0/,PTEMP/igen*0.D0/
      DATA PFRACV/igen*0.D0/,PENTH/igen*0.D0/
      DATA PRHO/igen*0.D0/
      data DEN/igen*0.D0,igen*0.D0,igen*0.D0,igen*0.D0,igen*0.D0/
c
      data reflag/.true./
C
      DATA INP/'.inp'/,ER1/'.er1'/
      DATA SCD/'.scd'/,TR2/'.tr2'/
      data scl/'.scl'/
C
C.....................................................
```

```
C
C     MAIN
C
C*** GET THE EXECUTION TIME
C
      t1 = secnds(0.)
      istat = lib$date_TIME(TSRC)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c     WRITE(lunlog,1130)
c1130 FORMAT(5X,'Enter the file name being used for this run: ',$)
      read(5,1000) nchar,opnrup   ! unit 5 gets command file too
 1000 format(q,40a1)
c
      opnrup1 = opnrup1(1:nchar) // er1(1:4)
c
      CALL ESTRT1(OPNRUP1)
      HTCUTS = HTCUT
C
      opnrup1 = opnrup1(1:nchar) // inp(1:4)
      CALL IO(tend,gmass0,OPNRUP1)
      if(check4) srcoer = 2.8*srcoer
      CALL ALPH
C
      alpha1 = alpha+1.
      WRITE(lunlog,1105) ALPHA
 1105 FORMAT(5X,'THE VALUE OF ALPHA IS ',F6.4)
C
C
      GAMMAF = GAMMA(1./ALPHA1)
      TSC1 = TEND
C
C
c*** set the density and enthalpy functions in TPROP
c
      call setenthal(h_masrte,h_airrte,h_watrte)
      call setden(1.D00,0.D00,h_masrte)
C
C.......................................................
C
C     SOURCE INTEGRATION (CA = RHOE)
C
      opnrup1 = opnrup1(1:nchar) // scd(1:4)
c
      OPEN(UNIT=9,NAME=OPNRUP1,recl=202,TYPE='SCRATCH',
     $  carriagecontrol='list',
     $  recordtype='variable')
c
      gmass  = gmass0
```

```
C
C...........................................................
C
C     START THE GAS BLANKET?
C
      L = 2.*AFGEN2(PTIME,R1T,TIME0,'R1T-MN')
      QSTRE = AFGEN2(PTIME,ET,TIME0,'ET-MN')/(pi*L**2/4.)
      PWCP = AFGEN2(PTIME,PWC,TIME0,'PWC')
      HPRIM = AFGEN2(PTIME,PENTH,TIME0,'PENTH')
      RHOP = AFGEN2(PTIME,PRHO,TIME0,'PRHO')
      CCP = PWCP*RHOP
C
C
      qstar = 0.
      if(u0 .ne. 0.)
     $       qstar = CCP*k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L)
C
C
      write(lunlog,3010) time0,l,sz,qstar,qstre
      IF(QSTRE.lt.QSTaR .and. smass0.eq.0.) then
            tsc1 = 0.
            GOTO 105
            endif
C
  100 CONTINUE
      check3 = .false.
C
C*** INITIAL CONDITIONS
C
      if(time0.ne. 0.) then
            L0 = 2.*AFGEN2(PTIME,R1T,TIME0,'R1T-MN')
            QSTRE = AFGEN2(PTIME,ET,TIME0,'ET-MN')/(pi*L0**2/4.)
      PWCP = AFGEN2(PTIME,PWC,TIME0,'PWC')
      HPRIM = AFGEN2(PTIME,PENTH,TIME0,'PENTH')
      RHOP = AFGEN2(PTIME,PRHO,TIME0,'PRHO')
      CCP = PWCP*RHOP
C
      qstar =0.
      if(u0.ne. 0.)
     $       qstar = CCP*k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L0)
C
            endif
C
C*** SET UP INTEGRATION PARAMETERS
C
C*** VARIABLE            SUBSCRIPT
C*** --------            ---------
C***  RG      Y(1)
C***  mass    Y(2)
C***  massc   Y(3)
C***  massa   Y(4)
```

```
C***  Enthalpy      Y(5)
c***  moe    y(6)
C***  TIME   X
C
      PRMT(1) = TIMEO
      PRMT(2) = 6.023E23
      PRMT(3) = STPIN
      PRMT(4) = ERBND
      PRMT(5) = STPMX
C     PRMT(6) = EMAX -- OUTPUT
c
      do iii = 6,25
      prmt(iii) = 0.
      enddo
C
      Y(1) = AFGEN2(PTIME,R1T,TIMEO,'R1T-MN')
      rmax = y(1)
      Y(2) = dmax1( gmass, (pi*Y(1)**2*1.1*srccut*RHOP))
      vuflag = .false.
      htod = gmass/RHOP/2./pi/Y(1)**3
      prmt(22) = htod * 2.*Y(1)   ! initial height of the tail
      prmt(24) = prmt(22)
      prmt(23) = 0.         ! initial height of the head
      prmt(25) = prmt(23)
      if(htod .gt. 0.1) vuflag = .true.
      Y(3) = Y(2) * PWCP
      PWAP = (1. - PWCP)/(1. + HUMID)       ! water from humidity only
      Y(4) = Y(2) * PWAP
      Y(5) = Y(2) * HPRIM
      y(6) = 0.0
C
      DERY(1) = WTRG
      DERY(2) = WTtm
      DERY(3) = WTyc
      DERY(4) = WTya
      DERY(5) = WTeb
      dery(6) = wtmb
C
      NDIM = 6
C
C*** PERFORM INTEGRATION
C
      WRITE(lunlog,1145)
 1145 FORMAT(5X,'Beginning Integration Step - Gas blanket')
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SRC1,SRC1O,AUX)
C
      IF(IHLF .GT. 10) CALL trap(1,IHLF)
      IF(CHECK3) then
            TEND = TSC1
            GO TO 110
      end if
C
C.........................................................
C
C     RESTART THE GAS BLANKET?
C
      TIME0 = TSC1
      L = 2.*AFGEN2(PTIME,R1T,TIME0,'R1T-MN')
      QSTRE = AFGEN2(PTIME,ET,TIME0,'ET-MN')/(pi*L**2/4.)
      PWCP = AFGEN2(PTIME,PWC,TIME0,'PWC')
      RHOP = AFGEN2(PTIME,PRHO,TIME0,'PRHO')
      CCP = PWCP*RHOP
c
      qstar = 0.
      if(u0.ne. 0.)
     $        qstar = CCP*k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L)
c
      write(lunlog,3010) time0,l,sz,qstar,qstre
 3010 format(//,' time0: ',1pg13.5,t40,'l: ',1pg13.5,
     $/,' sz0: ',1pg13.5,/,' qstar: ',1pg13.5,t40,'qstre: ',1pg13.5)


c
c*** before restarting, check to see if this may be a problem child.
c      If the current time is less than SRCSS, this may be a problem
c      simulation.  The scheme is to scan the primary source flux for the
c      rest of time.  If the flux is never higher, disable the blanket
c      restart feature in NOBL with the logical flag REFLAG, and
c      force the source to use NOBL.
c
       IF(QSTRE .Gt. QSTAR .and. time0.ge.srcss) then
             goto 100

       else if(QSTRE .GT. QSTAR) then

             do iii=1,isen
             if(ptime(iii) .eq. poundn) goto 100
             if(time0 .lt. ptime(iii)) goto 101
             enddo
             goto 100

 101         do ii=iii,isen
             if(et(ii) .eq. 0.D0) goto 102
             flux = et(ii)/(pi*r1t(ii)**2)
             if(flux .gt. qstre) goto 100
             enddo

 102         reflag = .false.
             write(lunlog,1146)
 1146        FORMAT(' Forcing NOBL...')
       endif
```

```
C.......................................................
C
C     SOURCE INTEGRATION -- NO GAS BLANKET
C
  105 continue
      WRITE(lunlog,1147)
 1147 FORMAT(5X,'Source calculation - No Gas blanket')
C
      CALL NOBL(timeout, reflag)
c
      if(check3) then        ! restart blanket calculation
            time0 = timeout
            goto 100
      end if
C
C
  110 RMAX = 1.01*RMAX      ! GUARANTEE A GOOD VALUE
      aleph = 0.
      if(u0 .ne. 0.)
     $              ALEPH = U0/GAMMAF*(SZM/Z0)**ALPHA
     $              /(SQRTPI/2.*RM +RMAX)**(ALPHA/ALPHA1)
C
c
      rewind (unit=9)
      opnrup1 = opnrup1(1:nchar) // scl(1:4)
      open(unit=8,name=opnrup1, type='new',
     $      carriagecontrol='fortran', recordtype='variable')
c
      call head(gmass0)
      call crfg(table,ntab,crfger)
      call head(gmass0)
c
      CLOSE(UNIT=9)
      close(unit=8)
C
      opnrup1 = opnrup1(1:nchar) // tr2(1:4)
      CALL TRANS(OPNRUP1)
C
C*** CALCULATE EXECUTION TIME
C
      tt1 = t1
      t1 = Secnds(tT1)/60.
      WRITE(lunlog,2000) TSRC
      WRITE(lunlog,2010) T1
 2000 FORMAT(1X,'BEGAN AT ',A24)
 2010 FORMAT(5X,' ***** ELAPSED TIME ***** ',1pg13.5,' min ')
C
      STOP
      END
$$$$
```

```
C..................................................
C
C
C     DIMENSIONS/DECLARATIONS for DEGADIS2
C
      include 'sys$degadis:DEGADIS1.dec/list'
C
C   MAXNOB IS THE MAXIMUM NUMBER OF OBSERVERS ALLOWED.
C
      parameter(    maxnob = 50,
     $              RT2= 1.41 421 3562D0,    ! sqrt(2.0)
     $              sqpio2= 1.25 331 4137D0)    ! sqrt(pi/2.)
C
$$$$
```

```
      PROGRAM DEGADIS2
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C     Program description:
C
C     DEGADIS2 performs the downwind dispersion portion of the calculation
C     for each of several observers released successively over the transient
C     gas source described by DEGADIS1.
C
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled 'Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
C
C     Disclaimer:
C
C     This computer code material was prepared by the University of
C     Arkansas as an account of work sponsored by the U. S. Coast Guard
C     and the Gas Research Institute.  Neither the University of Arkansas,
C     nor any person acting on its behalf:
C
C     a.  Makes any warranty or representation, express or implied,
C         with respect to the accuracy, completeness, or usefulness
C         of the information contained in this computer code material,
C         or that the use of any apparatus, method, numerical model,
C         or process disclosed in this computer code material may not
C         infringe privately owned rights; or
C
C     b.  Assumes any liability with respect to the use of, or for
C         damages resulting from the use of, any information,
C         apparatus, method, or process disclosed in this computer
C         code material.
```

```
C
C
C*************************************************************************
C*************************************************************************
C*************************************************************************
C
C
C
C
C
C
C     DIMENSIONS/DECLARATIONS
C
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec/list'
      include '($ssdef)'
C
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/SYCER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
     $/STOPIT/ TSTOP
     $/CNOBS/ NOBS
C
      character*80 TITLE(4)
C
      character*4 pound
      character*24 TINP,TSRC,TOBS,TSRT
      character*3 gas_name
C
      real*4 tt1
      REAL*8 K,ML,L
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      character*4 TR2,ER2,PSD,TR3, obs
      character OPNRUP(40)
      character*40 OPNRUP1
      equivalence (opnrup(1),opnrup1)
C
C.........................................................
C
C     DATA
C
      DATA TSTOP/0./
      DATA POUND/'//  '/,POUNDN/-1.E-20/
C
      DATA TIME0/0./,NDIM/0/
      DATA RADG/max12*0./,QSTR/max12*0./,srcden/max12*0./
      DATA NREC/maxnob*0,maxnob*0/,T0/maxnob*0./,XV/maxnob*0./
C
      DATA TR2/'.TR2'/,ER2/'.ER2'/
      DATA PSD/'.PSD'/,TR3/'.TR3'/, obs/'.OBS'/
C
C.........................................................
C
C     MAIN
C
      T1 = SECNDS(0.)
      istat = lib$date_TIME(TOBS)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c     WRITE(5,1130)
c1130 FORMAT(' Enter the file name being used for this run: ',$)
      read(5,1130) nchar,opnrup
 1130 format(q,40a1)
C
      opnrup1 = OPNRUP1(1:nchar) // er2(1:4)
      CALL ESTRT2(OPNRUP1)
C
C*** GET THE COMMON VARIABLES CARRIED FROM DEGADIS1
C
      opnrup1 = OPNRUP1(1:nchar) // tr2(1:4)
      CALL STRT2(OPNRUP1,H_masrte)
C
C
C.........................................................
```

```
C
C      PSEUDO STEADY STATE CALCULATIONS
C      INTEGRATION IN SUBROUTINE SUPERVISOR
C
       opnrup1 = OPNRUP1(1:nchar) // psd(1:4)
       OPEN(UNIT=9,TYPE='NEW',NAME=OPNRUP1,
     $   carriagecontrol='list',
     $   recordtype='variable')
C
       opnrup1 = OPNRUP1(1:nchar) // obs(1:4)
       OPEN( UNIT=12, TYPE='NEW',NAME=OPNRUP1,
     $   carriagecontrol='list',
     $   recordtype='variable')
C
       CALL SSSUP(H_masrte)
C
C.............................................................
C
       CLOSE(UNIT=9)
C
C
       call setden(1.D0,0.D0,H_masrte)     ! adiabatic mixing w/ pure stuff
C
C
       opnrup1 = OPNRUP1(1:nchar) // tr3(1:4)
       CALL TRANS(OPNRUP1)
C
       tt1 = t1
       T1 = SECNDS(tT1)
       T1 = T1/60.
       WRITE(lunlog,4000) TOBS
       WRITE(lunlog,4010) T1
 4000  FORMAT(3X,'BEGAN AT ',A40)
 4010  FORMAT(3X,'*** ELAPSED TIME *** ',1pG13.5,' min')
C
       STOP
       END
$$$$
```

```
C.....................................................
C
C     declarations for DEGADIS3
C
      include 'sys$degadis:DEGADIS2.dec/list'
C
      parameter (   maxnt=40,
     $              maxtnob=maxnt*maxnob)
C
####
```

```
      PROGRAM DEGADIS3
C
C*************************************************************************
C*************************************************************************
C*************************************************************************
C
C     Program description:
C
C     DEGADIS3 sorts the downwind dispersion calculation made for each of
C     the several observers in DEGADIS2.  The output concentrations at
C     several given times may then be corrected for along-wind dispersion
C     as desired.
C
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled 'Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
C
C     Disclaimer:
C
C     This computer code material was prepared by the University of
C     Arkansas as an account of work sponsored by the U. S. Coast Guard
C     and the Gas Research Institute.  Neither the University of Arkansas,
C     nor any person acting on its behalf:
C
C     a.  Makes any warranty or representation, express or implied,
C         with respect to the accuracy, completeness, or usefulness
C         of the information contained in this computer code material,
C         or that the use of any apparatus, method, numerical model,
C         or process disclosed in this computer code material may not
C         infringe privately owned rights; or
C
C     b.  Assumes any liability with respect to the use of, or for
C         damages resulting from the use of, any information,
C         apparatus, method, or process disclosed in this computer
```

```
C         code material.
C
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C
C
C


      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS3.dec/list'
      include '($ssdef)'
C
C*** MINIMUM DIMENSION ON TABLE IS 6 * MAXNOB + 1
C
      parameter (ntab0=10*maxnob+1)
C
      COMMON
     $/SORT/ TCA(maxnob,maxnt),TCASTR(maxnob,maxnt),
     $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $      Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      REAL*8 ML,K
      DIMENSION TABLE(ntab0)
C
      character*24 tsrc,tinp,tobs,tsrt
C
      character*3 gas_name
      character*4 TR3,PSD,Er3,SR3,Tr4
      character*40 opnrup1
      character opnrup(40)
C
      EQUIVALENCE (OPNRUP(1),opnrup1)
C.........................................................
C
C     DATA
C
      DATA POUNDN/-1.E-20/,POUND/'//  '/
      DATA TCA/maxtnob*0./,TCASTR/maxtnob*0./,TSY/maxtnob*0./
      data TSZ/maxtnob*0./,KSUB/maxnt*0/
      DATA TB/maxtnob*0./,TDIST0/maxtnob*0./,TDIST/maxtnob*0./
C
      DATA TR3/'.TR3'/,PSD/'.PSD'/
      DATA Er3/'.Er3'/,SR3/'.SR3'/,Tr4/'.Tr4'/
C
C*** UNITS
C*** 8 -- OUTPUT TO A PRINT FILE
C*** 9 -- I/O WITH DISK
C
      T1 = SECNDS(0.)
      istat = lib$date_time(tsrt)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c     WRITE(5,1130)
c1130 FORMAT(' Enter the file name used for this run: ',$)
      read(5,1130) nchar,opnrup
 1130 format(q,40a1)
C
C*** GET THE VERSION NUMBER
C
c 100 WRITE(5,1140)
c1140 FORMAT(' Enter the version number (between 00 and 99) for',
c    $' this sort: ',$)
c     CALL GTLIN(DUMMY)
c     NCAR = LEN(DUMMY)
c     IF(NCAR .EQ. 0) GO TO 110
C
c     IF(IVERIF(DUMMY,STRING) .NE. 0) GO TO 110
c     IF(NCAR-2) 130,140,120
C
c 110 WRITE(5,1150)
c1150 FORMAT(' ?DEGADIS3? - Invalid character for version number')
c     GO TO 100
C
c 120 WRITE(5,1160)
c1160 FORMAT(' ?DEGADIS3? - Too many characters in the version number')
c     GO TO 100
```

```
C
c 130 DOT(1) = '060
c     DOT(2) = DUMMY(1)
c     GO TO 150
C
c 140 DOT(1) = DUMMY(1)
c     DOT(2) = DUMMY(2)
C
c 150 CONTINUE
c     CALL CONCAT(Er3,DOT,Er3)
c     CALL CONCAT(Sr3,DOT,Sr3)
c     CALL CONCAT(Tr4,DOT,Tr4)
C
C*** NOW, REPLACE THE FILE NAME IN OPNRUP
C
c     CALL SCOPY(BFILE,OPNRUP)
C
C*** THATS IT
C
      opnrup1 = opnrup1(1:nchar) // tr3(1:4)
      CALL STRT3(OPNRUP1)
C
      opnrup1 = opnrup1(1:nchar) // er3(1:4)
      CALL ESTRT3(OPNRUP1)
C
      opnrup1 = opnrup1(1:nchar) // psd(1:4)
      OPEN(UNIT=9,NAME=OPNRUP1,TYPE='OLD')
C
C.......................................................................
C
C     TIME SORT SUPERVISOR -- CALCULATE DOWNWIND DISPERSION CORRECTION
C
      CALL SORTS(TABLE)
C
      CLOSE(UNIT=9)
C
C.....................................................................
C
C     OUTPUT SORTED PARAMETERS
C
      opnrup1 = opnrup1(1:nchar) // SR3(1:4)
      CALL SRTOUT(OPNRUP1, table)
C
      opnrup1 = opnrup1(1:nchar) // tr4(1:4)
      CALL TRANS(OPNRUP1)
C
      STOP
      END
####
```

```
  C.........................................................
  c
  c     declarations for DEGADIS4
  c
        include 'sys$degadis:DEGADIS3.dec/list'
  c
        parameter (ndos=10)
  c
####
```

```
      PROGRAM DEGADIS4
C
C***************************************************************************
C***************************************************************************
C***************************************************************************
C
C     Program description:
C
C     DEGADIS4 sorts the downwind dispersion calculation made for each of
C     the several observers in DEGADIS2.  The output concentrations at
C     several given times may then be corrected for along-wind dispersion
C     as desired.  DEGADIS4 also outputs the concentration as a function
C     of time at a given position.
C
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled 'Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
C
C     Disclaimer:
C
C     This computer code material was prepared by the University of
C     Arkansas as an account of work sponsored by the U. S. Coast Guard
C     and the Gas Research Institute.  Neither the University of Arkansas,
C     nor any person acting on its behalf:
C
C     a.  Makes any warranty or representation, express or implied,
C         with respect to the accuracy, completeness, or usefulness
C         of the information contained in this computer code material,
C         or that the use of any apparatus, method, numerical model,
C         or process disclosed in this computer code material may not
C         infringe privately owned rights; or
C
C     b.  Assumes any liability with respect to the use of, or for
C         damages resulting from the use of, any information,
C        apparatus, method, or process disclosed in this computer
C        code material.
C
C
C*************************************************************************
C*************************************************************************
C*************************************************************************
C
c
C
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS4.dec/list'
      include '($ssdef)'
C
C*** MINIMUM DIMENSION ON TABLE IS 6 * MAXNOB + 1
C
      parameter (ntab0=10*maxnob+1)
C
      COMMON
     $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $      Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/cdos/ idos, dosdisx(ndos), dosdis(4,2,ndos)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      REAL*8 ML,K
      DIMENSION TABLE(ntab0)
c
      character*24 tsrc,tinp,tobs,tsrt
C
```

```
      character*3 gas_name
      character*4 TR3,PSD,Er3,SR4,Tr4
      character*40 opnrup1
      character opnrup(40)
C
      EQUIVALENCE (OPNRUP(1),opnrup1)
C..................................................
C
C     DATA
C
      DATA POUNDN/-1.E-20/,POUND/'//  '/
      DATA TCc/maxtnob*0./,TCcSTR/maxtnob*0./,TSY/maxtnob*0./
      data TSZ/maxtnob*0./,KSUB/maxnt*0/
      DATA TB/maxtnob*0./,TDIST0/maxtnob*0./,TDIST/maxtnob*0./
C
      DATA TR3/'.TR3'/,PSD/'.PSD'/
      DATA Er3/'.Er3'/,SR4/'.SR4'/,Tr4/'.Tr4'/
C
C*** UNITS
C*** 8 -- OUTPUT TO A PRINT FILE
C*** 9 -- I/O WITH DISK
C
      T1 = SECNDS(0.)
      istat = lib$date_time(tsrt)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
      WRITE(lunlog,1130)
1130  FORMAT(' Enter the file name used for this run: ',$)
      read(5,1131) nchar,opnrup
 1131 format(q,40a1)
C
C
      opnrup1 = opnrup1(1:nchar) // tr3(1:4)
      CALL STRT3(OPNRUP1)
C
      opnrup1 = opnrup1(1:nchar) // er3(1:4)
      CALL ESTRT3(OPNRUP1)
C
c
c.....................................
c
c     input the position to calculate the concentration histograms
c
      write(6,*) 'Enter the number of downwind distances desired:'
      write(6,*) ' max of ',ndos,
     1     ' downwind distances; 4 positions at each distance'
      read(5,*) jdos
      if(jdos .gt. ndos) jdos=ndos
      do ii=1,jdos
      write(6,*) ' '
      write(6,*) 'enter the x coordinate:'
      read(5,*) dosdisx(ii)
            write(6,*)
      1      '  enter the y and z coordinate pairs at this distance:'
            do ij=1,4
            read(5,*) (dosdis(ij,jj,ii),jj=1,2)
            if(dosdis(ij,2,ii).le.0. .and.
      1      dosdis(ij,1,ii).le.0.) goto 100
            enddo
 100  continue
      enddo
c
c
      opnrup1 = opnrup1(1:nchar) // SR4(1:4)
      open(unit=8,name=opnrup1, type='new',carriagecontrol='fortran')

      do idos=1,jdos
C
C.....................................................................
C
C     TIME SORT SUPERVISOR -- CALCULATE DOWNWIND DISPERSION CORRECTION
C
      opnrup1 = opnrup1(1:nchar) // psd(1:4)
      OPEN(UNIT=9,NAME=OPNRUP1,TYPE='OLD')
c
      do ij=1,maxnt
      KSUB(ij) = 0
      do ijk=1,maxnob
      TSZ(ijk,ij) = 0.
      enddo
      enddo
c
c
      CALL SORTS(TABLE)
C
      CLOSE(UNIT=9)
C
C...................................................................
C
C     OUTPUT SORTED PARAMETERS
C
      CALL dosOUT(table)
c
      enddo
c
      close(unit=8)
c
C
```

```
      opnrup1 = opnrup1(1:nchar) // tr4(1:4)
      CALL TRANS(OPNRUP1)
C
      STOP
      END
####
```

```
C..............................................................
C
c     declarations for DEGADISIN
C
C
      parameter(    isen= 30,   ! dimension of /sen1/ and /sen2/
     $              pi= 3.14 159 2654D0)
c
####
```

```
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C     PROGRAM DEGADISIN
C
C     Program description:
C
C     DEGADISIN acts as an interactive input module to the programs
C     which make up the DEGADIS model.  The user is guided through a
C     series of questions which supply the model with the necessary
C     input information.
C
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled 'Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
C
C     Disclaimer:
C
C     This computer code material was prepared by the University of
C     Arkansas as an account of work sponsored by the U. S. Coast Guard
C     and the Gas Research Institute.  Neither the University of Arkansas,
C     nor any person acting on its behalf:
C
C     a.  Makes any warranty or representation, express or implied,
C         with respect to the accuracy, completeness, or usefulness
C         of the information contained in this computer code material,
C         or that the use of any apparatus, method, numerical model,
C         or process disclosed in this computer code material may not
C         infringe privately owned rights; or
C
C     b.  Assumes any liability with respect to the use of, or for
C         damages resulting from the use of, any information,
C         apparatus, method, or process disclosed in this computer
```

```
C        code material.
C
C
C*************************************************************************
C*************************************************************************
C*************************************************************************
C
C
C
C     INITIAL INPUT FOR DEGADIS ROUTINES
C
c     note: this series of programs relies on the system wide
c           logical symbol SYS$DEGADIS which denotes the source
c           and executable code for these images.
C
c
      PROGRAM DEGADISIN

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'SYS$DEGADIS:degadisin.dec'
c
      COMMON
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
C
      character*80 TITLE(4)
C
      character*3 gas_name
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
C
      REAL*8 ML,K
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
c check1
c check2=t  cloud type release with no liquid source; SRC1  DEGADIS1
c again      local communications in SSSUP          SSSUP
c check3    local communications between SRC1 and NOBL DEGADIS1
c check4=t  steady state simulation            DEGADISIN
```

```
c check5=t  operator sets sort parameters            ESTRT3
c
      data CHECK1/.false./,CHECK2/.false./,AGAIN/.false./
      data CHECK3/.false./,CHECK4/.false./,CHECK5/.false./
C
      character*100 OPNRUP
      character OPNRUP1(100)
      equivalence (opnrup1(1),opnrup)
      character*4 INP,er1,er2,er3,com,scl,sr3,lis
      character*4 dummy
      character*3 plus
      character*2 con
      DATA POUND/'// '/,POUNDN/-1.E-20/
C
      DATA PTIME/isen*0.D0/
      DATA ET/isen*0.D0/,R1T/isen*0.D0/
      DATA PWC/isen*0.D0/,PTEMP/isen*0.D0/
      DATA PFRACV/isen*0.D0/,PENTH/isen*0.D0/
      DATA PRHO/isen*0.D0/
      data DEN/isen*0.,isen*0.,isen*0.,isen*0.,isen*0./
      DATA INP/'.INP'/,er1/'.er1'/,er2/'.er2'/,er3/'.er3'/
      data scl/'.scl'/,sr3/'.sr3'/,lis/'.lis'/
      data com/'.com'/
      data plus/' + '/,con/' -'/
c
C*** GET THE FILE NAME TO BE USED BY ALL OF THE ROUTINES
C
      WRITE(6,800)
      WRITE(6,810)
      READ(5,820) NCHAR,opnrup
      opnrup = opnrup(1:nchar) // inp(1:4)
C
C*** NOW GET THE REST OF THE DESIRED INFORMATION
C
      CALL IOT(OPNRUP)
      WRITE(6,1000)
      if(check4) then
      write(6,1001) ! continuous
      else
            if(u0 .eq. 0.) then
             write(6,1009)
            else
             WRITE(6,1002) ! transient
            endif
      endif
      write(6,1010)
C
C*** FORMATS
C
  800 FORMAT(//,16X,'DEnse GAs DISpersion Model input module.')
  810 FORMAT(/,' Enter the simulation name',
     $' : [DIR]RUNNAME ',$)
  820 FORMAT(Q,A40)
 1000 FORMAT(' ',/,
     $' In addition to the information just obtained,',
     $' DEGADIS',/,' requires a series of numerical parameter',
     $' files which use',/,' the',
     $' same name as [DIR]RUNNAME given above. ',//,
     $'    For convenience, example parameter files are included for',/,
     $' each step.  They are:')
 1001 FORMAT(10X,'EXAMPLE.ER1 and',/,10X,'EXAMPLE.ER2')
 1002 format(10X,'EXAMPLE.ER1,',/,10X,'EXAMPLE.ER2, and',/,
     $10X,'EXAMPLE.ER3')
 1009 format(10x,'EXAMPLE.ER1')
 1010 format('  Note that each of',
     $' these files can be edited during the course of the',/,
     $' simulation if a parameter proves to be out of specification.',/)
c
c
      write(6,1200)
 1200 format(' Do you want a command file to be generated to execute',
     $' the procedure? <Y or n> ',$)
      REad(5,1210) dummy
 1210 format(a4)
      if(dummy.eq.'n' .or. dummy.eq.'N') goto 3000
      opnrup = opnrup(1:nchar) // com(1:4)
      write(6,1220) opnrup
 1220 format(' The command file will be generated under the file',
     $' name:',/,10x,a40)
c
      open(unit=8,name=opnrup,type='new',
     $ carriagecontrol='list',recordtype='variable')
c
      opnrup = opnrup(1:nchar) // er1(1:4)
c
      write(8,1250) (opnrup1(i),i=1,nchar+4)
 1250 format('$ copy/log SYS$DEGADIS:example.er1 ',40a1)
      IF(u0 .eq. 0.) then
            write(8,1280)
            write(8,1290) (opnrup1(i),i=1,nchar)
            goto 1340
            endif
      opnrup = opnrup(1:nchar) // er2(1:4)
c
      write(8,1260) (opnrup1(i),i=1,nchar+4)
 1260 format('$ copy/log SYS$DEGADIS:example.er2 ',40a1)
      opnrup = opnrup(1:nchar) // er3(1:4)
c
      if(.not.check4) then  ! transient
c
            write(8,1270) (opnrup1(i),i=1,nchar+4)
 1270       format('$ copy/log SYS$DEGADIS:example.er3 ',40a1)
```

```
c
           write(8,1280)
 1280      format('$ run SYS$DEGADIS:DEGADIS1')
           write(8,1290) (opnrup1(i),i=1,nchar)
 1290      format(40a1)
           write(8,1300)
 1300      format('$ run SYS$DEGADIS:DEGADIS2')
           write(8,1290) (opnrup1(i),i=1,nchar)
           write(8,1320)
 1320      format('$ run SYS$DEGADIS:DEGADIS3')
           write(8,1290) (opnrup1(i),i=1,nchar)
c
           opnrup = opnrup(1:nchar) // scl(1:4) //
     1     plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
           write(8,1370) (opnrup1(i),i=1,2*nchar+13)
 1370      format('$ copy/log ',100a1)
c
           opnrup = opnrup(1:nchar) // lis(1:4)
           write(8,1390) (opnrup1(i),i=1,nchar+4)
 1390      format('  ',40a1)
      else
           write(8,1280)
           write(8,1290) (opnrup1(i),i=1,nchar)
c
           write(8,1330)
 1330      format('$ run SYS$DEGADIS:SDEGADIS2')
           write(8,1290) (opnrup1(i),i=1,nchar)
c
           opnrup = opnrup(1:nchar) // scl(1:4) //
     1     plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
           write(8,1370) (opnrup1(i),i=1,2*nchar+13)
           opnrup = opnrup(1:nchar) // lis(1:4)
           write(8,1390) (opnrup1(i),i=1,nchar+4)
      endif
c
 1340 close(unit=8)
      write(6,1350)
 1350 format(/,' Do you wish to initiate this procedure? ',
     $' <y or N> ',$)
      REad(5,1210) dummy
      if(dummy.eq.'y' .or. dummy.eq.'Y') goto 2000
      goto 3000
 2000 opnrup = '@' // opnrup(1:nchar) // ' '
      istat = lib$do_command(opnrup)
      write(6,2100)
 2100 format(/,' ?DEGADISIN? command file failed to start.')
c
 3000 continue
      CALL EXIT
      END
$$$$

 5 -- SYS$DEGADIS:DEGADISIN.FOR               20-OCT-1987 00:16:35
```

```
      SUBROUTINE dosOUT(table)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS4.dec/list'
C
      COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $      Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $      TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/cdos/ idos, dosdisx(ndos), dosdis(4,2,ndos)
     $/SORTIN/TIM(maxnt),NTIM,ISTRT
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/alp/ alpha,alpha1
C
      dimension table(1)
      dimension chist(4,maxnt)
C
      logical cflag,cflag1
C
      character*3 gas_name
C
      if(sigx_flag.eq. 0.) then
            write(8,1102)
      else
            write(8,1104)
            write(8,1105) sigx_coeff,sigx_pow,sigx_min_dist
            endif
C
      cflag = isofl.eq. 1.or. ihtfl.eq. 0
      cflag1= isofl.eq.1
      if(cflag) then
            call adiabat(2,wc,wa,gas_lfl,ya,cc_lfl,r,w,t,tt)
            call adiabat(2,wc,wa,gas_ufl,ya,cc_ufl,r,w,t,tt)
            endif
C
      WRITE(8,1110) dosdisx(idos)
C
      WRITE(8,1119)
      WRITE(8,1119)
      if(cflag1) then
            WRITE(8,1116) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1118)
      else
            WRITE(8,1115) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1117)
            endif
      WRITE(8,1119)
      ip = 0
c
      DO 110 I=ISTRT,NTIM
C
      II = KSUB(I)
c
      j = 1
      disto  = tdist(j,i)
      cco    = tccstr(j,i)
      rhoo   = Trho(j,i)
      yco    = Tyc(j,i)
      tempo  = Ttemp(j,i)
      gammao = Tgamma(j,i)
      bo     = tb(j,i)
      szo    = tsz(j,i)
      syo    = tsy(j,i)
      if( disto .gt. dosdisx(idos) ) then
            write(lunlog,*) ' Extrapolated point for time: ',Tim(i)
      write(8,*) ' Records for ',tim(i),' s are missing - see source'
            goto 110
            endif
c
      DO 120 J=2,II
c
      dist  = tdist(j,i)
      cc    = tccstr(j,i)
      rho   = Trho(j,i)
      yc    = Tyc(j,i)
      temp  = Ttemp(j,i)
      gamma = Tgamma(j,i)
      b     = tb(j,i)
      sz    = tsz(j,i)
      sy    = tsy(j,i)
      blfl  = 0.
      bufl  = 0.

      if( dist .lt. dosdisx(idos) ) goto 119
      dist  = (dosdisx(idos) - disto)/(dist-disto)
      cc    = dist*(cc-cco) + cco
      rho   = dist*(rho-rhoo) + rhoo
      yc    = dist*(yc-yco) + yco
      temp  = dist*(temp-tempo) + tempo
      gamma = dist*(gamma-gammao) + gammao
      b     = dist*(b-bo) + bo
      sz    = dist*(sz-szo) + szo
      sy    = dist*(sy-syo) + syo
c
      if(.not.cflag) then
      call adiabat(-2,wc,wa,gas_lfl,ya,cc_lfl,r,w,gamma,tt)
      call adiabat(-2,wc,wa,gas_ufl,ya,cc_ufl,r,w,gamma,tt)
      endif
c
c*** calc the concentration time histories
c
      do ij=1,4
            chist(ij,i) = 0.
         if(dosdis(ij,2,idos) .ge. 0.) then
      arg = (dosdis(ij,2,idos)/sz)**alpha1
      if(dosdis(ij,1,idos) .gt. b) arg = arg +
      1              ((dosdis(ij,1,idos)-b)/sy)**2
            if(arg .lt. 80.) then
             chist(ij,i) = cc/exp(arg)
      if(cflag) then
            call adiabat(0,xx,aa,ycc,yaa,chist(ij,i),rr,ww,gg,tt)
      else
            call adiabat(-1,xx,aa,ycc,yaa,chist(ij,i),rr,ww,gamma,tt)
      endif
            chist(ij,i) = ycc
            endif
         endif
      enddo
c
c
      arg = (gas_zsp/sz)**alpha1
      if(arg .ge. 80.) goto 600
c
      ccz = cc/exp(arg)
      if(ccz .lt. cc_lfl) then
            if(cflag1) then
      WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY
             else
      WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY
             endif
            goto 600
            endif
      arg = -(dlog(cc_lfl/cc) + (gas_zsp/sz)**alpha1)
      blfl = sqrt(arg)*sy + b
c
      if(ccz .lt. cc_ufl) then
            if(cflag1) then
      WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY,blfl
             else
      WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
             endif
            goto 600
            endif
      arg = -(dlog(cc_ufl/cc) + (gas_zsp/sz)**alpha1)
      bufl = sqrt(arg)*sy + b
            if(cflag1) then
```

```
        WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY,blfl,bufl
                else
        WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl,bufl
                endif
c
  600 continue
        goto 120
  119 continue
        disto  = tdist(j,i)
        cco    = tccstr(j,i)
        rhoo   = Trho(j,i)
        yco    = Tyc(j,i)
        tempo  = Ttemp(j,i)
        gammao = Tgamma(j,i)
        bo     = tb(j,i)
        szo    = tsz(j,i)
        syo    = tsy(j,i)
  120 CONTINUE
c
        ip = ip + 1
        if(ip .eq. 3) then
                ip = 0
                write(8,1119)
                endif
  110 CONTINUE
C
c*** output the concen time histories
c
        write(8,1119)
        write(8,1119)
        iii = 0
        do ij=1,4
           if(dosdis(ij,2,idos).gt.0. .and.
        1       dosdis(ij,1,idos).gt.0.) iii = ij
        enddo

        if(iii .eq. 0) return
        ip = 0

        write(8,2100) ((dosdis(ij,iij,idos),ij=1,4),iij=1,2)
 2100   format(1H0,11x,'Time',11x,4(4x,'Mole fraction at:',5x),/,
        1     1H ,26x,4(4x,'y= '1pg13.5,' m',4x),/,
        1     1H ,11x,'(s)',12x,4(4x,'z= '1pg13.5,' m',4x))
        DO I=ISTRT,NTIM
        write(8,2200) tim(i),(chist(ij,i),ij=1,iii)
        ip = ip + 1
        if(ip .eq. 3) then
                ip = 0
                write(8,1119)
                endif
        enddo
 2200 format(1h ,5(6x,1pg14.7,6x))
C
C
 1102 format(1H0,5x,'X-Direction correction was NOT applied.')
 1104 format(1H0,5x,'X-Direction correction was applied.')
 1105 format(1h ,5x,5x,'Coefficient:      ',1pg13.5,/,
     1     1h ,5x,5x,'Power:            ',1pg13.5,/,
     1     1h ,5x,5x,'Minimum Distance: ',1pg13.5' m')
 1110 FORMAT(1H0,5X,'Centerline values for the position -->',/,
     1     ' x: ',1pg13.5,' m',/)
 1115 FORMAT(1H0,1X,'  Time  ',2x,3x,'Mole',3x,
     1     'Concentration',1x,'Density',2x,3x,'Gamma',3x,
     1     'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
     1     'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
     1     11x,11x,11x,11x,3x,'Width',3x,11x,9x,
     1     2(1pg9.3,'mole%',1x))
 1116 FORMAT(1H0,1X,'  Time  ',2x,3x,'Mole',3x,
     1     'Concentration',1x,'Density',2x,
     1     'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
     1     'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
     1     11x,11x,11x,3x,'Width',3x,11x,9x,
     1     2(1pg9.3,'mole%',1x))
 1117 FORMAT(1H ,4X,'(s)',4x,11x,
     1     2(1X,'(kg/m**3)',1x),11x,4x,'(K)',
     1     5(8X,'(m)'))
 1118 FORMAT(1H ,4X,'(s)',4x,11x,
     1     2(1X,'(kg/m**3)',1x),4x,'(K)',
     1     5(8X,'(m)'))
 1119 FORMAT(1H )
 1120 FORMAT(1H ,3(1X,1PG9.3,1X),2x,0pf7.4,2x,1X,1PG10.3,1X,
     1     6(1X,1PG9.3,1X))
C
      RETURN
      END
####
```

```
C..............................................................
C
C      ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
       SUBROUTINE ESTRT1(OPNRUP)

       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
       include 'sys$degadis:DEGADIS1.dec'
C
       parameter(    iend= 22,
       1      iend1= iend+1,
       1      iiiend= 2,
       1      iiiend1= iiiend+1,
       1      iiend= 2,
       1      iiend1= iiend+1,
       1      jend= 4,
       1      jend1= jend+1,
       1      mend= 5,
       1      mend1= mend+1)
C
C      BLOCK COMMON
C
       COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfser,epsilon
     $/vucom/ vua,vub,vuc,vud,vudelta,vuflag
     $/szfc/ szstp0,szerr,szstpmx,szsz0
     $/alphcom/ ialpfl,alpco
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
C
       EQUIVALENCE
     $(RLBUF(1),STPIN),   !MAIN - RKGST - INITIAL STEP SIZE
     $(RLBUF(2),ERBND),   !MAIN - RKGST - ERROR BOUND
     $(RLBUF(3),STPMX),   !MAIN - RKGST - MAXIMUM STEP SIZE
     $(RLBUF(4),WTRG),    !MAIN - RKGST - WEIGHT FOR RG
     $(RLBUF(5),WTtm),    !MAIN - RKGST - WEIGHT FOR Total mass
     $(RLBUF(6),WTya),    !MAIN - RKGST - WEIGHT FOR Ya
     $(RLBUF(7),WTyc),    !MAIN - RKGST - WEIGHT FOR Yc
     $(RLBUF(8),WTeb),    !MAIN - RKGST - WEIGHT FOR Energy Balance
     $(RLBUF(9),WTmb),    !MAIN - RKGST - WEIGHT FOR Momentum Balance
     $(RLBUF(10),WTuh),   !MAIN - RKGST - WEIGHT FOR Ueff*Heff
     $(RLBUF(11),XLI),    !ALPH - LOWER LIMIT OF SEARCH FOR ALPHA
     $(RLBUF(12),XRI),    !ALPH - UPPER LIMIT OF SEARCH FOR ALPHA
     $(RLBUF(13),EPS)     !ALPH - ERROR BOUND USED BY 'RTMI'
       equivalence
     $(RLBUF(14),ZLOW),   !ALPHI - BOTTOM HEIGHT FOR FIT OF ALPHA
     $(RLBUF(15),STPINZ), !ALPHI - INITIAL RKGST STEP <0.
     $(RLBUF(16),ERBNDZ), !ALPHI - ERROR BOUND FOR RKGST
     $(RLBUF(17),STPMXZ), !ALPHI - MAXIMUM STEP FOR RKGST
     $(RLBUF(18),SRCOER), !SRC1O - OUTPUT error criteria
     $(RLBUF(19),SRCSS),  !SRC1O - min time for Steady;4*STPMX
     $(RLBUF(20),SRCcut), !SRC1O - min height for blanket
     $(RLBUF(21),htcut),  !SRC1 - min height for blanket heat transfer
     $(RLBUF(iend),ERNOBL), !NOBL - CONVERGENCE CRITERIA
     $(RLBUFi(1),crfger), !CRFG - Error criteria for building tables
     $(RLBUFi(iiiend),epsilon)!SRC1 - Coefficient in Air entrainment
C
      equivalence
     $(rlbufa(1),ce),       !SRC1 - Coefficient gravity slumping EQ
     $(RLBUFa(iiend),delrhomin) ! stop spread for delrho<delrhomin
C
      equivalence
     $(rlbuf1(1),szstp0),  ! SZF - Initial step size
     $(rlbuf1(2),szerr),   ! SZF - Error criteria
     $(rlbuf1(3),szstpmx), ! SZF - Maximum step size
     $(rlbuf1(4),szsz0)    ! SZF - Initial value of rho*dellay*UHeff
C
C
      equivalence
     $(rlbuf4(1),vua),     ! Constant Av in SRC1
     $(rlbuf4(2),vub),     ! Constant Bv in SRC1
     $(rlbuf4(3),vuc),     ! Constant Ev in SRC1
     $(rlbuf4(4),vud),     ! Constant Dv in SRC1
     $(rlbuf4(5),vudelta)  ! Constant DELTAv in SRC1
C
      character*40 OPNRUP
      character DUMMY(1:132)
      DIMENSION RLBUF(iend), rlbufi(iiiend), rlbufa(iiend)
      dimension rlbuf1(jend)
      dimension rlbuf4(mend)
C
      logical vuflag
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD',err=2000)
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
      I = 1
  100 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 100
      DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
      I = I + 1
      if(i .eq. iend1) goto 110
      GO TO 100

  110 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 110
      DECODE(20,1010,DUMMY,ERR=400) ptnobl
      NOBLPT = INT(PTNOBL)

      I = 1
  120 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 120
      DECODE(20,1010,DUMMY,ERR=400) RLBUFi(I)
      I = I + 1
      if(i .eq. iiend1) goto 140
      GO TO 120
C
C*** READ A LINE AND DETERMINE ITS PURPOSE for /sprd_con/
C
  140 I = 1
  150 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 150
      DECODE(20,1010,DUMMY,ERR=400) RLBUFa(I)
      I = I + 1
      if(i .eq. iiend1) goto 190
      GO TO 150
C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill szfc
C
  190 I = 1
  200 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 200
      DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
      I = I + 1
      if(i .eq. jend1) goto 230
      GO TO 200
C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /alphcom/
C
  230 I = 1
  240 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 240
      DECODE(20,1010,DUMMY,ERR=400) Ralpfl

      ialpfl = int(ralpfl)

  250 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 250
      DECODE(20,1010,DUMMY,ERR=400) alpco
C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /phicom/
```

```
C
  260 I = 1
  270 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 270
      DECODE(20,1010,DUMMY,ERR=400) Rphifl

      iphifl = int(rphifl)

  275 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 275
      DECODE(20,1010,DUMMY,ERR=400) dellay
c
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /vucom/
C
  280 I = 1
  290 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 290
      DECODE(20,1010,DUMMY,ERR=400) RLBUF4(I)
      I = I + 1
      if(i .eq. mend1) goto 300
      GO TO 290
c
C*** EXIT THE PROCEEDINGS
C
  300 CONTINUE
      CLOSE(UNIT=9)
      RETURN
c
  350 call trap(20)
C
  400 CONTINUE
      CALL trap(21)
C
 1000 FORMAT(Q,132A1)
 1010 FORMAT(10X,G10.4)
C
 2000 call trap(22)
      END
$$$$
```

```
C..............................................................
C
C      ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
      SUBROUTINE ESTRT2(OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS2.dec/list'
c
      parameter (   ienda= 18,
      1      iendal= ienda+1,
      2      iendb= 7,
      3      iendb1= iendb+1)
c
      common
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/STP/STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/CNOBS/NOBS
C
      EQUIVALENCE
     $(RLBUF(1),SYOER),    !SSSUP - RKGST - INITIAL SY
     $(RLBUF(2),ERRO),     !SSSUP - RKGST(OBS) - ERROR BOUND
     $(RLBUF(3),SZOER),    !SSSUP - RKGST(OBS) - INITIAL SZ
     $(RLBUF(4),WTAIO),    !SSSUP - RKGST(OBS) - WEIGHT FOR AI
     $(RLBUF(5),WTQOO),    !SSSUP - RKGST(OBS) - WEIGHT FOR Q
     $(RLBUF(6),WTSZO),    !SSSUP - RKGST(OBS) - WEIGHT FOR SZ
     $(RLBUF(7),ERRP),     !SSSUP - RKGST(PSS) - ERROR BOUND
     $(RLBUF(8),SMXP),     !SSSUP - RKGST(PSS) - MAXIMUM STEP
     $(RLBUF(9),WTSZP),    !SSSUP - RKGST(PSS) - WEIGHT FOR SZ
     $(RLBUF(10),WTSYP),   !SSSUP - RKGST(PSS) - WEIGHT FOR SY
     $(RLBUF(11),WTBEP),   !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
     $(RLBUF(12),WTDH),    !SSSUP - RKGST(PSS) - WEIGHT FOR DH
     $(RLBUF(13),ERRG),    !SSSUP - RKGST(SSG) - ERROR BOUND
     $(RLBUF(14),SMXG),    !SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
     $(RLBUF(15),ERTDNF),  !TDNF - CONVERGENCE CRITERIA
     $(RLBUF(16),ERTUPF),  !TUPF - CONVERGENCE CRITERIA
     $(RLBUF(17),WTruh),   !SSSUP - RKGST(SSG) - WEIGHT FOR RUH
     $(RLBUF(ienda),WTdhg)!SSSUP - RKGST(SSG) - WEIGHT FOR DH
C
      EQUIVALENCE
     $(RLBUF1(1),STPO),    !SSSUP - RKGST(OBS) - INITIAL STEP
     $(RLBUF1(2),STPP),    !SSSUP - RKGST(PSS) - INITIAL STEP
     $(RLBUF1(3),ODLP),    !SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
     $(RLBUF1(4),ODLLP),   !SSSUP - RKGST(PSS) - MAXIMUM DISTANCE TO OUT
     $(RLBUF1(5),STPG),    !SSSUP - RKGST(SSG) - INITIAL STEP
     $(RLBUF1(6),ODLG),    !SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
     $(RLBUF1(iendb),ODLLG)!SSSUP- RKGST(SSG) - MAXIMUM DISTANCE TO OUT
C
```

```
      character*40 OPNRUP
      character dummy(1:132)
      DIMENSION RLBUF(ienda),RLBUF1(iendb)
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** FIRST, FILL RLBUF
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
      I = 1
  100 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 100
      DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
      I = I + 1
      IF(I .EQ. ienda1) GO TO 200
      GO TO 100
C
C*** NOW, FILL RLBUF1
C
  200 I = 1
  210 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 210
      DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
      I = I + 1
      IF(I .EQ. iendb1) GO TO 260
      GO TO 210
C
C*** NOW, PICK UP NOBS
C
  260 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 260
      DECODE(20,1010,DUMMY,ERR=400) RBUF
      NOBS = INT(RBUF)
C
C*** EXIT THE PROCEEDINGS
C
      CLOSE(UNIT=9)
      RETURN
C
  300 call trap(20)
  400 CALL trap(21)
C
 1000 FORMAT(Q,132A1)
 1010 FORMAT(10X,G10.4)
      END
$$$$
```

```
C.............................................................
C
C     ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
      SUBROUTINE ESTRT2SS(OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
      include 'sys$degadis:DEGadis2.dec/list'
c
      parameter(    ienda= 18,
      1       iendal= ienda+1,
      2       iendb= 7,
      3       iendb1= iendb+1)
C
      COMMON
     $/ERROR/SYOER,ERRP,SMXP,WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,
     $ WTRUH,WTDHG
     $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C
c
       character*40 OPNRUP
       character DUMMY(1:132)
       DIMENSION RLBUF(ienda),RLBUF1(iendb)
C
       OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** FIRST, FILL RLBUF
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
       I = 1
  100 CONTINUE
       READ(9,1000,END=350) NCHAR,DUMMY
       IF(DUMMY(1) .EQ. '!') GO TO 100
       DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
       I = I + 1
       IF(I.EQ. iendal) GOTO200
       GO TO 100
C
C*** NOW, FILL RLBUF1
C
  200 i = 1
  210 READ(9,1000,END=350) NCHAR,DUMMY
       IF(DUMMY(1) .EQ. '!') GO TO 210
       DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
       I = I + 1
       if(i.eq. iendb1) goto 300
       GO TO 210
C
```

```
C*** EXIT THE PROCEEDINGS
C
  300 CONTINUE
      sy0er = rlbuf(1)  !SSSUP - RKGST - INITIAL SY
      errp  = rlbuf(7)  !SSSUP - RKGST(PSS) - ERROR BOUND
      smxp  = rlbuf(8)  !SSSUP - RKGST(PSS) - MAXIMUM STEP
      wtszp = rlbuf(9)  !SSSUP - RKGST(PSS) - WEIGHT FOR SZ
      wtsyp = rlbuf(10) !SSSUP - RKGST(PSS) - WEIGHT FOR SY
      wtbep = rlbuf(11) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
      wtDH  = rlbuf(12) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
      errs  = rlbuf(13) !SSSUP - RKGST(SSG) - ERROR BOUND
      smxs  = rlbuf(14) !SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
      wtRUH = rlbuf(17) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
      wtDHG = rlbuf(18) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
C
      stpp  = rlbuf1(2) !SSSUP - RKGST(PSS) - INITIAL STEP
      odlp  = rlbuf1(3) !SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
      odllp = rlbuf1(4) !SSSUP - RKGST(PSS) - MAXIMUM DISTANCE TO OUT
      stps  = rlbuf1(5) !SSSUP - RKGST(SSG) - INITIAL STEP
      odls  = rlbuf1(6) !SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
      odlls = rlbuf1(7) !SSSUP - RKGST(SSG) - MAXIMUM DISTANCE TO OUT
C
      CLOSE(UNIT=9)
      RETURN
C
  350 call trap(20) ! premature EOF
C
  400 CALL trap(21)
C
 1000 FORMAT(Q,132A1)
 1010 FORMAT(10X,G10.4)
C
      END
$$$$
```

```
C.......................................................
C
C      ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
      SUBROUTINE ESTRT3(OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      COMMON
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ERROR/ERT1,ERDT,ERNTIM
C
      EQUIVALENCE
     $(RLBUF(1),ERT1),     !FIRST SORT TIME - USER OPTION
     $(RLBUF(2),ERDT),     !SORT TIME DELTA - USER OPTION
     $(RLBUF(3),ERNTIM)    !NUMBER OF SORT TIMES - USER OPTION
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      character DUMMY(1:132)
      character*40 opnrup
      DIMENSION RLBUF(3),RBUF(6)
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
      I = 1
  100 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 100
      DECODE(20,1010,DUMMY,ERR=400) RBUF(I)
      I = I + 1
      GO TO 100
C
C*** EXIT THE PROCEEDINGS AND DETERMINE CHECK5
  300 CONTINUE
C
      DO 310 I = 1,3
  310 RLBUF(I) = RBUF(I)
      CHECK5 = .FALSE.        ! IN ORDER FOR FLAG TO WORK
      IF(RBUF(4) .EQ. 1.) CHECK5 = .TRUE.
C
      sigx_flag = rbuf(5)
      CLOSE(UNIT=9)
      RETURN
C
  400 CALL trap(21)
 1000 FORMAT(Q,132A1)
 1010 FORMAT(10X,G10.4)
      END
$$$$

 1 -- SYS$DEGADIS:ESTRT3.FOR                     20-OCT-1987 00:11:14
```

```
C
C     ..............................................................
C
C     SUBROUTINE GAMMA
C
C..............................................................
C
C     This routine was originally supplied by Digital Equipment
C     Corporation as part of the Scientific Subroutine Package
C     available for RT-11 as part of the Fortran Enhancement
C     Package.  It was upgraded for use in this package.
C
C..............................................................
C
C     PURPOSE
C           COMPUTES THE GAMMA FUNCTION FOR A GIVEN ARGUMENT
C
C     USAGE
C           GF = GAMMA(XX)
C
C     DESCRIPTION OF PARAMETERS
C           XX -THE ARGUMENT FOR THE GAMMA FUNCTION
C
C     IER-RESULTANT ERROR CODE WHERE
C           IER=0  NO ERROR
C           IER=1  XX IS WITHIN .000001 OF BEING A NEGATIVE INTEGER
C           IER=2  XX GT 34.5, OVERFLOW
C           IF IER .NE. 0 PROGRAM TAKES A DIP IN THE POOL!
C
C     REMARKS
C           NONE
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C     METHOD
C           THE RECURSION RELATION AND POLYNOMIAL APPROXIMATION
C           BY C.HASTINGS,JR., 'APPROXIMATIONS FOR DIGITAL COMPUTERS',
C           PRINCETON UNIVERSITY PRESS, 1955
C
C     MODIFIED TO FUNCTION FORM FROM ORIGINAL SUBROUTINE FORM
C
C     ..................................................................
C
      FUNCTION GAMMA(XX)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      IF(XX-34.5)6,6,4
    4 IER=2
      GAMMA=1.E38
      GO TO 1000
    6 X=XX
      ERR=1.0E-6
      IER=0
      GAMMA=1.0
      IF(X-2.0)50,50,15
   10 IF(X-2.0)110,110,15
   15 X=X-1.0
      GAMMA=GAMMA*X
      GO TO 10
   50 IF(X-1.0)60,120,110
C
C        SEE IF X IS NEAR NEGATIVE INTEGER OR ZERO
C
   60 IF(X-ERR)62,62,80
   62 Y=FLOAT(INT(X))-X
      IF(ABS(Y)-ERR)130,130,64
   64 IF(1.0-Y-ERR)130,130,70
C
C        X NOT NEAR A NEGATIVE INTEGER OR ZERO
C
   70 IF(X-1.0)80,80,110
   80 GAMMA=GAMMA/X
      X=X+1.0
      GO TO 70
  110 Y=X-1.0
      GY=1.0+Y*(-0.5771017+Y*(+0.9858540+Y*(-0.8764218+Y*
     $(+0.8328212+Y*(-0.5684729+Y*(+0.2548205+Y*(-0.05149930)))))))
      GAMMA=GAMMA*GY
  120 RETURN
  130 IER=1
 1000 CONTINUE
      IF(IER.EQ.1) WRITE(5,1100)
      IF(IER.EQ.2) WRITE(5,1110)
 1100 FORMAT(5X,'?GAMMA?--ARGUMENT LESS THAN 0.000001')
 1110 FORMAT(5X,'?GAMMA?--ARGUMENT GREATER THAN 34.5--OVERFLOW')
      CALL EXIT
      END
####
```

```
C..............................................................
C
C     SUBROUTINE TO ESTABLISH THE TIME SORT PARAMETERS
C
      SUBROUTINE GETTIM

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
      LOGICAL CHECK1,CHECK2,CHECK3,CHECK4,CHECK5,AGAIN
C
      DATA T1/0./,DT/0./,TF/0./
C
C*** IF CHECK5 IS SET, GET THE TIME SORT PARAMETERS FROM /ERROR/
C
      IF(.NOT.CHECK5) GO TO 90
C
      T1 = ERT1
      DT = ERDT
      NTIM = INT(ERNTIM)
      GO TO 95
C
C
C*** This subroutine sets the default time sort windows.
C
C*** The first sort time is set for potential low wind speed cases,
C***   while the last sort time is set for potential high wind speed
C***   cases.  The first sort time is taken to be when the first
C***   observer passes through x=RMAX.  The last sort time is taken
C***   to be when the last observer passes through x=6*RMAX.
C***   The default value for the number of sort times is set to 10.
C***   Obviously, these values generate some sort times which will be
C***   useless; hopefully, these values will show the user where to
C***   look on the next go around.
C*** The number of times has been doubled to 20 and the time interval
C***   has been doubled in order to give more information for lower
C***   concentrations (for the toxic gas problem). tos,5mar86
C
  90  CONTINUE
C
```

```
      T1 = TO(1) + (2.*RMAX)**(1./ALPHA1)/ALEPH
      TF = TO(NOBS) + (6.*RMAX)**(1./ALPHA1)/ALEPH
      NTIM = 20            ! for toxic gas problem
C
c DT = (TF-T1)/FLOAT(NTIM-1)
      DT = 2.*(TF-T1)/FLOAT(NTIM-1) ! for toxic gas problem
      DT = FLOAT(INT(DT+.5))
C
      IF(DT .GE. 5.) GO TO 95
      DT = 5.
      NTIM = INT((TF - T1)/DT) + 1
C
   95 CONTINUE
C
      T1 = FLOAT(INT(T1))  !MAKE T1 AN INTEGER VALUE
C
      II = max( ntim,2)    ! fill in the lowest two times at least
      DO 100 I = 1,II
      TIM(I) = DT*FLOAT(I-1) + T1
  100 CONTINUE
C
      RETURN
      END
$$$$
```

```
C.............................................................
C
C     SUBROUTINE TO ESTABLISH THE TIME SORT PARAMETERS
C
      SUBROUTINE GETTIM

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS4.dec/list'
C
      COMMON
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/cdos/ idos, dosdisx(ndos), dosdis(4,2,ndos)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
      LOGICAL CHECK1,CHECK2,CHECK3,CHECK4,CHECK5,AGAIN
C
      DATA T1/0./,DT/0./,TF/0./
C
C*** This subroutine sets the time sort windows for concentration as
C     a function of time at a given position.
C
      dist = dosdisx(idos)
C
      T1 = ts( T0(1), dist) ! time first observer crosses dist
      TF = ts( T0(nobs), dist)    ! time last observer crosses dist
      NTIM = 40
C
      DT = (TF-T1)/FLOAT(NTIM-1)
      DT = dmax1( DBLE(INT(DT+.5)), 1.D0)
      NTIM = min( INT((TF - T1)/DT) + 1, 40)
C
      T1 = FLOAT(INT(T1))  !MAKE T1 AN INTEGER VALUE
      write(lunlog,1000) t1, tf, dt, dist
 1000 format('  t1: ',1pg13.5,'  tf: ',1pg13.5,'  dt: ',1pg13.5,
     $' dist:  ',1pg13.5)
C
      II = max( ntim,2)    ! fill in the lowest two times at least
      DO 100 I = 1,II
      TIM(I) = DT*FLOAT(I-1) + T1
  100 CONTINUE
C
      RETURN
      END
####
```

```
      SUBROUTINE HEAD(gmass0)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS1.dec'
      include '($ssdef)'
c
      COMMON
     $/GEN3/ RADG(2,maxl),QSTR(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/ STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/phlag/ check1,check2,again,check3,check4,check5
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
c
      character*80 TITLE(4)
c
      character*4 pound
      character*24 TINP,TSRC,TOBS,TSRT
      character*3 gas_name
      character*1 stabil(6)
      character*24 id
c
      logical check1,check2,again,check3,check4,check5
c
      REAL*8 K,ML
c
      data stabil/'A','B','C','D','E','F'/
      data iparm/0/
C
      if(iparm .eq. 1) goto 190
      WRITE(8,1100)
 1100 FORMAT(1H0,'********************',9X,'U O A _ D E G A D I S ',
     $2X,'M O D E L ',2X,'O U T P U T ',2X,'- - ',2X,'V E R S I O N ',
     $2X,'2.0',8X,'**********************')
C
C
      WRITE(8,1111)
C
      WRITE(8,1102) tsrc
 1102 FORMAT(1H ,'***************',23X,
     $'***************',1X,a24,1X,
     $'***************',23X,'***************')
C
      WRITE(8,1111)
C
      WRITE(8,1112) TINP
      WRITE(8,1114) TSRC
      IF(tOBS(1:2).NE.'  ' .and. .not.check4) WRITE(8,1116) TOBS
      IF(tOBS(1:2).NE.'  ' .and. check4) WRITE(8,1117) TOBS
      IF(tSRT(1:2) .NE. '  ') WRITE(8,1118) TSRT
 1112 FORMAT(1H ,'Data input on',22X,a24)
 1114 FORMAT(1H ,'Source program run on',14X,a24)
 1116 FORMAT(1H ,'Pseudo Steady-State program run on ',a24)
 1117 FORMAT(1H ,'Steady-State program run on',7x,a24)
 1118 FORMAT(1H ,'Time sort program run on',11X,a24)
      WRITE(8,1111)
C
      write(8,1119)
 1119 format(//,
     11h ,10x,22('*****'),/,
     21h ,10x,'*',t121,'*',/,
     31h ,10x,'*',t20,'NOTE:',t121,'*',/,
     21h ,10x,'*',t20,'-----',t121,'*',/,
     21h ,10x,'*',t121,'*',/,
     21h ,10x,'*',t20,'>',t25,'All Calculations are lim',
     3'ited to circular liquid sources.',t121,'*',/,
     21h ,10x,'*',t121,'*',/,
     11h ,10x,22('*****'),//)
      WRITE(8,1110)
      WRITE(8,1111)
C
 1110 FORMAT(1H0,10X,'TITLE BLOCK')
 1111 FORMAT(1H )
C
      DO 100 I = 1,4
      WRITE(8,1120) TITLE(I)
  100 CONTINUE
C
 1120 FORMAT(1H ,A80)
C
      WRITE(8,1111)
      WRITE(8,1130) U0
      WRITE(8,1140) Z0
      WRITE(8,1150) ZR
      write(8,1155) stabil(istab)
      if(ml.ne. 0.) then
           WRITE(8,1160) ML
      else
           write(8,1161)
      endif
      WRITE(8,1170) DELTA
      WRITE(8,1180) BETA
      WRITE(8,1190) ALPHA
      WRITE(8,1192) USTAR
      WRITE(8,1194) tamb
      if(isofl.eq.0 .and. ihtfl.ne.0) write(8,1195) tsurf
      WRITE(8,1196) pamb
      WRITE(8,1198) humid
      vaporp = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
      relhumid = 100.* humid/(0.622*vaporp / (pamb- vaporp))
      write(8,1199) relhumid
C
 1130 FORMAT(1H ,5X,'Wind velocity at reference height ',20X,F6.2,2X,
     $'m/s')
 1140 FORMAT(1H ,5X,'Reference height ',37X,F6.2,2X,'m')
 1150 FORMAT(1H0,5X,'Surface roughness length ',25X,1PG10.3,2X,'m')
 1155 FORMAT(1H0,5X,'Pasquill Stability class ',25X,4x,a1)
 1160 FORMAT(1H0,5X,'Monin-Obukhov length ',29X,1PG10.3,2X,'m')
 1161 FORMAT(1H0,5X,'Monin-Obukhov length ',31X,'infinite')
 1170 FORMAT(1H ,5X,'Gaussian distribution constants ',4X,'Delta',
     $10X,F9.5,2X,'m')
 1180 FORMAT(1H ,5X,32X,4X,'Beta',11X,F9.5)
 1190 FORMAT(1H0,5X,'Wind velocity power law constant',4X,'Alpha',
     $10X,F9.5)
 1192 FORMAT(1H ,5X,'Friction velocity',15X,4X,5X,10X,F9.5,2X,'m/s')
 1194 FORMAT(1H0,5X,'Ambient Temperature ',34X,F6.2,2X,'K')
 1195 FORMAT(1H0,5X,'Surface Temperature ',34X,F6.2,2X,'K')
 1196 FORMAT(1H ,5X,'Ambient Pressure ',37X,F6.3,2X,'atm')
 1198 FORMAT(1H ,5X,'Ambient Absolute Humidity',25X,1PG10.3,2X,
     $'kg/kg BDA')
 1199 FORMAT(1H ,5X,'Ambient Relative Humidity',25X,4X,F6.2,2X,'%')
C
      WRITE(8,1111)
C
      if(isofl .eq. 0) goto 135
      WRITE(8,1200)
      WRITE(8,1205)
      ii = -1
      DO 130 I = 1,igen
      IF(DEN(1,I).gt. 1.) goTO 148
      ii = ii+1
      if(ii.eq. 3) then
           write(8,1211)
           ii = 0
           endif
```

```
  130 WRITE(8,1210) DEN(1,I),DEN(2,I),den(3,i)
      goto 148
  135 write(8,1207)
      write(8,1208)
      ii = -1
      DO 138 I = 1,igen
      IF(DEN(1,I).gt. 1.) goTO 148
      ii = ii+1
      if(ii.eq. 3) then
            write(8,1211)
            ii = 0
            endif
  138 WRITE(8,1212) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
  148 continue
C
 1200 FORMAT(1H ,5X,'Input:    ',6X,3x,'Mole fraction',4x,
     1      'CONCENTRATION OF C',6X,'GAS DENSITY')
 1205 FORMAT(1H ,14X,20x,2(13X,'kg/m**3'))
 1207 FORMAT(1H ,5X,'Adiabatic Mixing:',3x,'Mole fraction',3x,
     1      'CONCENTRATION OF C',6X,'GAS DENSITY',5x,
     1      6x,'Enthalpy',6x,4x,'Temperature')
 1208 FORMAT(1H ,14X,20x,2(13X,'kg/m**3'),7x,8x,'J/kg',8x,9x,'K')
 1210 FORMAT(1H ,14X,3(12X,F8.5))
 1211 format(1H )
 1212 FORMAT(1H ,14X,3(12X,F8.5),6x,3x,1pg13.5,7x,1pg13.5)
C
      WRITE(8,1111)
      write(8,1233) gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp
      WRITE(8,1111)
      WRITE(8,1220) gmass0
      WRITE(8,1230)
C
      DO 150 I=1,IGEN
      IF(PTIME(I).EQ.POUNDN) GO TO 160
  150 WRITE(8,1231) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),
     $   PENTH(I)
  160 CONTINUE
C
 1220 FORMAT(1H ,'Source input data points',//,
     $        1h ,15x,'Initial mass in cloud: ',1pg13.5,//,
     $        1h ,7X,'Time',10X,'Contaminant',
     $        7X,'Source Radius',2x,3x,'Contaminant',4x,
     $        3x,'Temperature',4x,5x,'Enthalpy',5x,/,
     $        1x,18x,4x,'Mass Rate',5x,18x,2x,'Mass Fraction',3x)
 1230 FORMAT(1H ,9X,'s',15X,'kg/s',16X,'m',8x,' kg contam/kg mix ',8x,
     $        'K',9x,7x,'J/kg',7x)
 1231 FORMAT(1H ,6(3X,1PG12.5,3X))
 1233 Format(1H0,5x,'Specified Gas Properties:',//,
     110x,'Molecular weight:',T65,1pg13.5,/,
     110x,'Storage temperature:',T65,1pg13.5,'K',/,
      110x,'Density at storage temperature and ambient pressure:',
      1      T65,1pg13.5,'kg/m**3',/,
      110x,'Mean heat capacity constant:',T65,1pg13.5,/,
      110x,'Mean heat capacity power:',T65,1pg13.5,/,
      110x,'Upper mole fraction contour:',T65,1pg13.5,/,
      110x,'Lower mole fraction contour:',T65,1pg13.5,/,
      110x,'Height for isopleths:',T65,1pg13.5,'m',/)
 1241 format(1h0,5x,'Calculation procedure for ALPHA: ',I2)
 1242 format(1h0,5x,'Entrainment prescription for PHI: ',I2)
 1244 format(1h0,5x,'Layer thickness ratio used for average depth: ',
      1      1pg13.5)
 1250 format(1h0,5x,'Air entrainment coefficient used: ',f5.3)
 1251 format(1h0,5x,'NON Isothermal calculation')
 1252 format(1h0,5x,'Gravity slumping velocity coefficient used: ',f5.3)
 1253 format(1h0,5x,'Heat transfer calculated with fixed coefficient: ',
      1      1pg13.5,' J/m**2/s/K')
 1254 format(1h0,5x,'Heat transfer not included')
 1255 format(1h0,5x,'Heat transfer calculated with correlation: ',I2)
 1256 format(1h0,5x,'Isothermal calculation')
 1257 format(1h0,5x,'Water transfer calculated with fixed coefficient: ',
      1      1pg13.5,' /m**2/s/atm')
 1258 format(1h0,5x,'Water transfer not included')
 1259 format(1h0,5x,'Water transfer calculated with correlation')
C
      WRITE(8,1111)
      write(8,1241) ialpfl
      write(8,1242) iphifl
      write(8,1244) dellay
      write(8,1250) epsilon
      write(8,1252) ce
      if(isofl.eq. 0) write(8,1251)
      if(isofl.ne. 0) write(8,1256)
      if(ihtfl.lt. 0) write(8,1253) htco
      if(ihtfl.eq. 0) write(8,1254)
      if(ihtfl.gt. 0) write(8,1255) ihtfl
      if(iwtfl.lt. 0) write(8,1257) wtco
      if(iwtfl.eq. 0) write(8,1258)
      if(iwtfl.gt. 0) write(8,1259)
      WRITE(8,1111)
      iparm =1
      return
c
  190 continue
      if(.not.check4) return
      RAD = SQRT(2.*SLEN*SWID/pi)
      WRITE(8,1300) ESS,RAD
      WRITE(8,1320) SLEN,SWID
      qstar = ess/outl**2
      WRITE(8,1340) OUTCc,OUTSZ,qstar
      write(8,1350) swcl,swal,senl,srhl
      WRITE(8,1360) OUTL,OUTB
```

```
C
C
C
 1300 FORMAT(1H0,'Source strength [kg/s] : ',18X,1PG13.5,T60,
     $'Equivalent Primary source radius [m] : ',5x,1PG13.5)
 1320 FORMAT(1H ,'Equivalent Primary source length [m] : ',4X,1PG13.5,
     $T60,'Equivalent Primary source half-width [m] : ',1X,1PG13.5)
 1340 FORMAT(/,' Secondary source concentration [kg/m**3] : ',
     $1PG13.5,T60,'Secondary source SZ [m] : ',18X,1PG13.5,//,
      1 ' Contaminant flux rate: ',1pg13.5,/)
 1350 format(/,' Secondary source mass fractions... contaminant: ',
      1     1pg13.6,2x,' air: ',1pg13.5,/,' ',10x,' Enthalpy: ',
      1     1pg13.5,5x,' Density: ',1pg13.5,/)
 1360 FORMAT(1H ,'Secondary source length [m] : ',13X,1PG13.5,T60,
     $'Secondary source half-width [m] : ',10X,1PG13.5)
C
C
      RETURN
      END
####
```

```
C.......................................................
c
c     gamma and incomplete gamma function
c
C.......................................................
c
c     routines included in this file:
c           GAMINC         Incomplete gamma function
c           GAMMLN         Natural log of the gamma function
c           GSER           computation procedure
c           GCF            computation procedure
c
C.......................................................
c
c     These functions return the value for the incomplete Gamma function
c     with the arguments (ALPHA,BETA).  The functional values are
c     calculated with either a series representation or a continued
c     fraction representation.  These routines are based on:
c
c     Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling,
c     'Numerical Recipes', Cambridge University Press, Cambridge,1986.
c
c
c
      function gaminc(alpha,beta)

      implicit Real*8 (a-h, o-z), integer*4 (i-n)
c
c
c*** ensure the arguments are within the proper bounds.
c
      if( beta.lt.0. .or. alpha.le.0.) then
            write(6,9000) alpha,beta
 9000       format(' GAMINC? Arugments are out-of-bounds. ALPHA: ',
     $ 1PG13.5,' BETA: ',1PG13.5)
            STOP
            endif
c
c*** determine which of the series or continued fraction representation
c    is more appropriate.
c
      if( beta .lt. alpha+1. ) then
c                                     ! series
            call gser( aa, alpha, beta, gln)
      else
c                                     ! continued fraction
            call gcf( aa, alpha, beta, gln)
            aa = 1. - aa
      endif
c
c*** multiply the result by GAMMA(ALPHA) to get the final value.
```

```
c
      gaminc = dexp( dlog(aa) + gln)
      return
      end
c
c
c
      subroutine gser( gamser, alpha, beta, gln)
c
c     This routine calculates the incomplete gamma function using
c     the series representation.
c
      implicit real*8 (a-h,o-z), integer*4 (i-n)
c
      parameter (itmax=100, eps=1.d-9)

      gln = gammln(alpha)

      if(beta .lt. 0.) then
            write(6,*) 'GSER? BETA is out-of-bounds.'
            stop
            endif
      if(beta .eq. 0.) then
            gamser = 0.
            return
            endif

      ap = alpha
      sum = 1./alpha
      del = sum

      do 100 n=1,itmax
            ap = ap + 1.
            del = del* beta /ap
            sum = sum + del
            if( dabs(del) .lt. dabs(sum)*eps) then
             gamser = sum*dexp( -beta+alpha*dlog(beta)-gln)
             return
             endif
 100  continue

      write(6,*) 'GSER? ALPHA is too large or ITMAX is too small.'
      stop
      end
c
c
c
      subroutine gcf( gamcf, alpha, beta, gln)
c
c     This routine calculates the incomplete gamma function using
c     the continued fraction representation.
```

```
c
      implicit real*8 (a-h,o-z), integer*4 (i-n)
c
      parameter (itmax=100, eps=1.d-9)

      gln = gammln(alpha)

c
c*** INITIALIZE some variables.
c
c                    ! previous value tested for convergence
      gold = 0.
      a0 = 1.
      a1 = beta
      b0 = 0.
      b1 = 1.
c                    ! fac is a 'renormalization factor'
      fac = 1.

      do 100 n=1,itmax
            an = dble(n)
            ana = an - alpha
            a0 = (a1 + a0*ana)
            b0 = (b1 + b0*ana)
            anf = an
            a1 = beta *a0 + anf*a1
            b1 = beta *b0 + anf*b1
            if( a1 .ne. 0.) then
             fac = 1./b1
             gg = a1*fac
            a0 = a0*fac
            a1 = a1*fac
            b0 = b0*fac
            b1 = b1*fac
             if( dabs((gg-gold)/gg) .lt .eps) then
                    gamcf = dexp(
     $                      -beta+alpha*dlog(beta)-gln)/gg
                    return
                    endif
             gold = gg
            endif
 100  continue

      write(6,*) 'GCF? ALPHA is too large or ITMAX is too small.'
      stop
      end
c
c
c
      function gammln( alpha )
c
```

```
c     This routine calculates ln( gamma(alpha)) for alpha>0.
c
      implicit real*8 (a-h,o-z), integer*4 (i-n)
c
      dimension cof(6)

      data cof,stp/76.18009173D0, -86.50532033D0, 24.01409822D0,
     $ -1.231739516D0, 0.120858003D-2, -0.536382D-5, 2.50662827465D0/
      data half, one, fpf/0.5D0, 1.0D0, 5.5D0/
c
      if( alpha .lt. 1) then
            gam = gamma( alpha )
            gammln = dlog(gam)
            return
            endif
c
      xx = alpha - one
      tmp = xx + fpf
      tmp = (xx + half) * dlog(tmp) - tmp
      ser = one
      do 100 j=1,6
            xx = xx + one
            ser = ser + cof(j)/xx
 100  continue

      gammln = tmp + dlog(stp*ser)
      return
      end
$$$$
```

```
C.......................................................
C
C      INPUT SUBROUTINE FOR DEGADIS MODEL
C
      SUBROUTINE IO(tend,gmass0,OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      include 'sys$degadis:DEGADIS1.dec'
C
C      BLOCK COMMON
C
      COMMON
     $/TITL/TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/phlag/check1,check2,again,check3,check4,check5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/POUNDN,POUND
     $/ICNTL/ IPRINT,IOBS(10)
C
C
      character*80 TITLE(4)
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
      character*3 gas_name
C
      REAL*8 ML,K
      logical check1,check2,again,check3,check4,check5
C
      character*40 OPNRUP
C
      OPEN(UNIT=9, NAME=OPNRUP, TYPE='OLD')
C
      DO 90 I=1,4
      READ(9,2000) TITLE(I)
   90 CONTINUE
 2000 FORMAT(A80)
C
      READ(9,*) U0,Z0,zr
      read(9,*) istab
      READ(9,*) DELTA,BETA,ml
      read(9,*) sigx_coeff,sigx_pow,sigx_min_dist
      read(9,*) tamb,pamb,humid
      humsrc = 0.
      read(9,*) isofl,tsurf
      read(9,*) ihtfl,htco
      read(9,*) iwtfl,wtco
      read(9,2020) gas_name
 2020 FORMAT(A3)
      read(9,*) gas_mw,gas_temp,gas_rhoe
      read(9,*) gas_cpk,gas_cpp
      read(9,*) gas_ufl,gas_lfl,gas_zsp
C
      if(isofl .eq. 0) then
            rhoe = gas_rhoe
            rhoa = pamb*(1.+humid)/(.002833+ 0.004553*humid)/tamb
            goto 105
            endif
      READ(9,*) NP
      DO 100 I=1,NP
  100 READ(9,*) DEN(1,I),DEN(2,I),den(3,I),den(4,I),den(5,i)
      RHOE = DEN(3,NP)
      RHOA = DEN(3,1)
      den(1,np+1) = 2.
C
  105 READ(9,*) CcLOW
C
      read(9,*) gmass0
      READ(9,*) NP
      DO 110 I=1,NP
      READ(9,*) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),PFRACV(I)
      WA = (1.D0 - PWC(I))/(1.D0 + HUMID)
      PENTH(I) = ENTHAL( PWC(I), WA, PTEMP(I))
      CALL TPROP(-1,PWC(I), WA,PENTH(I), YC,YA,WM, PTEMP(I), PRHO(I),CP)
 110  CONTINUE
      TEND = PTIME(NP-2)
      PTIME(NP + 1)  = POUNDN
C
      read(9,*) check1,check2,again,check3,check4,check5
c
      tobs = ' '
      tgrt = ' '
      READ(9,2010) TINP
 2010 format(a24)
C
      if(check4) read(9,*) ess,slen,swid
c
      CLOSE(UNIT=9)
      RETURN
      END
$$$$
```

```
      SUBROUTINE IOT(OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADISIN.dec'
C
      COMMON
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ UO,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
C
      character*80 TITLE(4)
      character*3 gas_name
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
C
      REAL*8 ML,K
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      character*(*) OPNRUP
      character*40 STRING
      character*4 dummy
C
      WRITE(6,1100)
      WRITE(6,1110)
C
C*** OPEN THE INPUT FILE
C
      OPEN(UNIT=8,NAME=OPNRUP,TYPE='NEW',
     $  carriagecontrol='list',recordtype='variable')
C
C*** NOW GET THE TITLE BLOCK
C
      WRITE(6,1120)
      WRITE(6,1130)
C
      DO 100 I=1,4
      READ(5,1134) TITLE(I)
      dummy = title(i)
      IF(dummy(1:4) .EQ. POUND(1:4)) GO TO 110
      WRITE(8,1135) TITLE(I)
```

```
  100 CONTINUE
      GO TO 130
C
  110 CONTINUE             ! FILL OUT THE BLOCK
      II = I
      DO 120 I = II,4
      TITLE(I) = ' '
      WRITE(8,1135) TITLE(I)
  120 CONTINUE
  130 CONTINUE
C
c*** Atmospheric parameters:
c
      WRITE(6,1140)
      WRITE(6,1142)
      READ(5,*) U0,Z0,ZR
      WRITE(8,1020) U0,Z0,ZR
C
c*** stability, averaging time for DELTA, and derived parameters
c
      WRITE(6,1150)
      READ(5,1310) NCHAR,STRING
      write(6,1152)
      read(5,*) avtime
      timeav = dmax1( avtime, 20.D0)
      istab = 4               ! default is D stability
      IF(STRING.eq.'A' .or. string.eq.'a') istab=1
      IF(STRING.eq.'B' .or. string.eq.'b') istab=2
      IF(STRING.eq.'C' .or. string.eq.'c') istab=3
      IF(STRING.eq.'D' .or. string.eq.'d') istab=4
      IF(STRING.eq.'E' .or. string.eq.'e') istab=5
      IF(STRING.eq.'F' .or. string.eq.'f') istab=6
      goto(161,162,163,164,165,166) istab
 161  delta = 0.224*(timeav/20.D0)**0.2D0       ! A
      if(timeav .gt. 600.D0) delta = 0.443*(timeav/600.D0)**0.5D0
      beta = 0.894
      ml = -11.43 * zr**0.103
      sigx_coeff = 0.02
      sigx_pow = 1.22
      sigx_min_dist = 130.
      goto 170
 162  delta = 0.164*(timeav/20.D0)**0.2D0       ! B
      if(timeav .gt. 600.D0) delta = 0.324*(timeav/600.D0)**0.5D0
      beta = 0.894
      ml = -25.98 * zr**0.171
      sigx_coeff = 0.02
      sigx_pow = 1.22
      sigx_min_dist = 130.
      goto 170
 163  delta = 0.109*(timeav/20.D0)**0.2D0       ! C
      if(timeav .gt. 600.D0) delta = 0.216*(timeav/600.D0)**0.5D0
      beta = 0.894
      ml = -123.4 * zr**0.304
      sigx_coeff = 0.02
      sigx_pow = 1.22
      sigx_min_dist = 130.
      goto 170
 164  delta = 0.071*(timeav/20.D0)**0.2D0        ! D
      if(timeav .gt. 600.D0) delta = 0.141*(timeav/600.D0)**0.5D0
      beta = 0.894
      ml = 0.0              ! used for infinity
      sigx_coeff = 0.04
      sigx_pow = 1.14
      sigx_min_dist = 100.
      goto 170
 165  delta = 0.053*(timeav/20.D0)**0.2D0        ! E
      if(timeav .gt. 600.D0) delta = 0.105*(timeav/600.D0)**0.5D0
      beta = 0.894
      ml = 123.4 * zr**0.304
      sigx_coeff = 0.17
      sigx_pow = 0.97
      sigx_min_dist = 50.
      goto 170
 166  delta = 0.036*(timeav/20.D0)**0.2D0        ! F
      if(timeav .gt. 600.D0) delta = 0.071*(timeav/600.D0)**0.5D0
      beta = 0.894
      ml = 25.98 * zr**0.171
      sigx_coeff = 0.17
      sigx_pow = 0.97
      sigx_min_dist = 50.
C
 170  WRITE(8,1040) istab
C
 172  if( ml.ne. 0.) then
      WRITE(6,1159) delta,beta,ml,sigx_coeff,sigx_pow,sigx_min_dist
      else
      WRITE(6,1160) delta,beta,sigx_coeff,sigx_pow,sigx_min_dist
      endif
      read(5,1310) nchar,string
      if(string.eq.'d' .or. string.eq.'D') then
            write(6,1600)
            read(5,*) delta
            goto 172
      else if(string.eq.'b' .or. string.eq.'B') then
            write(6,1620)
            read(5,*) beta
            goto 172
      else if(string.eq.'l' .or. string.eq.'L') then
            write(6,1660)
            read(5,*) ml
            goto 172
      else if(string.eq.'c' .or. string.eq.'C') then
```

```
                write(6,1670)
                read(5,*) sigx_coeff
                goto 172
          else if(string.eq.'p' .or. string.eq.'P') then
                write(6,1680)
                read(5,*) sigx_pow
                goto 172
          else if(string.eq.'m' .or. string.eq.'M') then
                write(6,1690)
                read(5,*) sigx_min_dist
                goto 172
          else if(nchar.eq.0 .or. string.eq.'n' .or. string.eq.'N') then
                WRITE(8,1020) DELTA,BETA,ml
                WRITE(8,1020) sigx_coeff,sigx_pow,sigx_min_dist
          else
          goto 172
          endif
c
c*** ambient pressure, temperatures, and humidity
c
          write(6,1500)
          read(5,*) tamb,pamb
          tamb = tamb + 273.15 ! K
          vaporp = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
          sat = 0.622*vaporp / (pamb- vaporp)
          write(6,1580)
          read(5,1310) nchar,string
          if(string.eq.'a' .or. string.eq.'A') then
                write(6,1585)
                read(5,*) humid
                relhumid = 100.*humid/sat
                write(6,1586) relhumid
                goto 200
                endif
          write(6,1587)
          read(5,*) relhumid
          humid = relhumid/100. * sat
    200   rhoa = pamb*(1.+humid)/(.002833+.004553*humid)/tamb
          write(6,1588) rhoa
          write(8,1025) tamb,pamb,humid
c
          isofl = 0
          ihtfl = 0
          htco  = 0.
          iwtfl = 0
          wtco  = 0.
          tsurf = tamb
c
          write(6,2000)
          read(5,1310) nchar,string
          if(string.eq.'Y' .or. string.eq.'y') then
```

```
            isofl = 1
            tsurf = tamb
            goto 250
            endif
c
      write(6,2020)
      read(5,1310) nchar,string
      if(string.eq.'Y' .or. string.eq.'y') then
            write(6,2030)
            read(5,*) tsurf
 220        write(6,2040)
            read(5,1310) nchar,string
             if(string.eq.'V' .or. string.eq.'v') then
             ihtfl = -1    ! constant value
             write(6,2050)
             read(5,*) htco
      else if(string.eq.'C' .or. string.eq.'c' .or. nchar.eq.0) then
                     ihtfl = 1     ! local correlation
             else if(string.eq.'L' .or. string.eq.'l') then
                     ihtfl = 2     ! LLNL correlation
                     htco = 0.0125 ! [=]m/s
                     write(6,2043) htco
                     read(5,1310) nchar,string
                     if(string.eq.'Y' .or. string.eq.'y')
     1                      read(5,*) htco
             else
             goto 220
             endif
      else
      goto 250
      endif
c
      write(6,2100)
      read(5,1310) nchar,string
      if(string.eq.'Y' .or. string.eq.'y') then
            iwtfl = 1
            write(6,2045)
            read(5,1310) nchar,string
             if(string.eq.'V' .or. string.eq.'v') then
             iwtfl = -1
             write(6,2120)
             read(5,*) wtco
             endif
            endif
c
 250  continue
      write(8,1060) isofl,tsurf
      write(8,1060) ihtfl,htco
      write(8,1060) iwtfl,wtco
c
C
```

```
c*** gas characteristics
c
      write(6,1510)
      read(5,1415) gas_name
      write(8,1415) gas_name
      gas_mw = 16.04
      gas_temp = 111.7
      gas_rhoe = 1.792*pamb ! correct to pamb
      gas_cpk = 2730.
      gas_cpp = 1.00
      gas_ufl= 0.15
      gas_lfl= 0.05
      gas_zsp= 0.5

      if(gas_name.eq.'NH3' .or. gas_name.eq.'nh3') then
            gas_mw = 17.
            gas_temp = tamb
            gas_rhoe = pamb*gas_mw/0.08205/tamb ! ideal gas
            gas_cpk = 3845.       ! to get 807.5 J/kg/K
            gas_cpp = 1.000
            gas_ufl= 2.0E-2
            gas_lfl= 2.00E-3
            gas_zsp= 0.5

      else if(gas_name.eq.'LNG' .or. gas_name.eq.'lng') then
            gas_mw = 16.04
            gas_temp = 111.7
            gas_rhoe = 1.792*pamb ! correct to pamb
            gas_cpk = 5.6e-8
            gas_cpp = 5.00
            gas_ufl= 0.15
            gas_lfl= 0.05
            gas_zsp= 0.5

      else if(gas_name.eq.'LPG' .or. gas_name.eq.'lpg') then
            gas_mw = 44.09
            gas_temp = 231.
            gas_rhoe = 2.400*pamb ! correct to pamb
            gas_cpk = 15.4
            gas_cpp = 2.25
            gas_ufl= 0.10
            gas_lfl= 0.02
            gas_zsp= 0.5

      else if(gas_name.eq.'NO2' .or. gas_name.eq.'no2') then
            gas_mw = 46.
            gas_temp = tamb
            gas_rhoe = pamb*gas_mw/0.08205/tamb ! ideal gas
            gas_cpk = 3845.       ! to get 807.5 J/kg/K
            gas_cpp = 1.000
            gas_ufl= 1000.0E-6
            gas_lfl= 500.00E-6
            gas_zsp= 0.5

      else if(gas_name.eq.'N2O' .or. gas_name.eq.'n2o') then
            gas_mw = 92.
            gas_temp = tamb
            gas_rhoe = pamb*gas_mw/0.08205/tamb ! ideal gas
            gas_cpk = 40990.      ! to get 807.5 J/kg/K
            gas_cpp = 1.000
            gas_ufl= 1000.0E-6
            gas_lfl= 500.00E-6
            gas_zsp= 0.5

      else if(gas_name.eq.'CL2' .or.
      1     gas_name.eq.'cl2' .or. gas_name.eq.'Cl2') then
            gas_mw = 70.91
            gas_temp = 238.7
            gas_rhoe = 3.672*pamb ! correct to pamb
            gas_cpk = 484.2
            gas_cpp = 1.000
            gas_ufl= 0.1D-4
            gas_lfl= 0.3D-5
            gas_zsp= 0.5

      else if(gas_name.eq.'MEC' .or.gas_name.eq.'mec') then
            gas_mw = 84.94
            gas_temp = tamb
            gas_rhoe = 3.53*(298.15/tamb)*pamb ! correct to pamb,tamb
            gas_cpk = 2730.       ! Open to question?
            gas_cpp = 1.000
            gas_ufl= 1000.D-6
            gas_lfl= 500.D-6
            gas_zsp= 0.5
      endif
  270 write(6,1520) gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
      1     gas_ufl,gas_lfl,gas_zsp
      read(5,1310) nchar,string
      if(string.eq.'m' .or. string.eq.'M') then
            write(6,1550)
            read(5,*) gas_mw
            goto 270
      else if(string.eq.'t' .or. string.eq.'T') then
            write(6,1530)
            read(5,*) gas_temp
            goto 270
      else if(string.eq.'d' .or. string.eq.'D') then
            write(6,1535)
            read(5,*) gas_rhoe
            goto 270
      else if(string.eq.'h' .or. string.eq.'H') then
            write(6,1570)
            read(5,*) gas_cpk
            goto 270
      else if(string.eq.'p' .or. string.eq.'P') then
            write(6,1571)
            read(5,*) gas_cpp
            goto 270
      else if(string.eq.'u' .or. string.eq.'U') then
            write(6,1572)
            read(5,*) gas_ufl
            goto 270
      else if(string.eq.'l' .or. string.eq.'L') then
            write(6,1573)
            read(5,*) gas_lfl
            goto 270
      else if(string.eq.'z' .or. string.eq.'Z') then
            write(6,1574)
            read(5,*) gas_zsp
            goto 270
      else if(nchar.eq. 0 .or. string.eq.'n' .or. string.eq.'N') then
            WRITE(8,1020) gas_mw, gas_temp,gas_rhoe
            write(8,1020) gas_cpk,gas_cpp
            WRITE(8,1020) gas_ufl,gas_lfl,gas_zsp
      else
      goto 270
      endif
c
c density curve if isothermal
c
      if(isofl .eq. 0) goto 460
      WRITE(6,1161)
      WRITE(6,1162)
      WRITE(6,1163)
      WRITE(6,1164) rhoa
      WRITE(6,1165)
      WRITE(6,1166)
      goto 320
C
 280  write(6,1290)
C
 320  LUNIN = 5
      WRITE(6,1300)
      READ(5,1310) NCHAR,STRING
      IF(STRING.EQ.'y' .or. string.eq.'Y') GO TO 360
      GO TO 400
  360 WRITE(6,1320)
      READ(5,1310) NCHAR,STRING
      OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=280)
      LUNIN = 10
  400 CONTINUE
      IF(LUNIN .EQ. 5) WRITE(6,1170) igen
      READ(LUNIN,*) NP
      WRITE(8,1040) NP
      IF(LUNIN .EQ. 5) WRITE(6,1180)
C
      DO 440 I=1,NP
      den(4,i) = 0.                 ! 0.0 by default for isotherm
      den(5,i) = tamb                ! tamb for isotherm
      READ(LUNIN,*) DEN(1,I),DEN(2,I),DEN(3,I)
      if(i .eq.1 .and.
     1     (den(3,1)/rhoa.gt.1.005 .or. rhoa/den(3,1).gt.1.005)) then
           den(3,1) = rhoa
           write(6,1340) rhoa
           endif
      if(i.eq.np) THEN
           IF(     den(2,i)/gas_rhoe .gt. 1.005
     1     .or.    gas_rhoe/den(2,i) .gt. 1.005
     1     .or.    den(3,i)/gas_rhoe .gt. 1.005
     1     .or.    gas_rhoe/den(3,i) .gt. 1.005) then
           den(2,i) = gas_rhoe
           den(3,i) = gas_rhoe
           write(6,1341) gas_rhoe
           endif
           endif
      WRITE(8,1025) DEN(1,I),DEN(2,I),DEN(3,I),Den(4,I),den(5,i)
  440 CONTINUE
      IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
C
c
 460  ccmax = (gas_lfl/2.) * gas_rhoe * (gas_temp/tamb)
      WRITE(6,1280) ccmax
      READ(5,*) CcLOW
      if(cclow .le. 0.) cclow=0.005 ! don't let 0. get through
      if(cclow .gt. ccmax) cclow=ccmax
      WRITE(8,1010) CcLOW
c
c
c*** source description
c
      write(6,1440)
c
      write(6,1460)
      read(5,1410) dummy
      if(dummy.eq.'d' .or. dummy.eq.'D') goto 730
c
      check4 = .false.
      write(6,1400)
      read(5,1410) dummy
      if(dummy.eq.'y' .or. dummy.eq.'Y') goto 480
      goto 520
 480  continue
      gmass0 = 0.           ! no initial cloud for a SS simulation
      write(8,1020) gmass0
      write(6,1420)
      read(5,*) ess
      write(6,1430)
      read(5,*) r1ss
      np = 4
c
      tend = 6023. ! [=] sec
c
      PTIME(1) = 0.
      et(1) = ess
      r1t(1)= r1ss
      pwc(1) = 1.
      ptemp(1) = gas_temp
      pfracv(1) = 1.
      PTIME(2) = tend
      et(2) = ess
      r1t(2)= r1ss
      pwc(2) = 1.
      ptemp(2) = gas_temp
      pfracv(2) = 1.
      PTIME(3) = tend + 1.
      et(3) = 0.
      r1t(3)= 0.
      pwc(3) = 1.
      ptemp(3) = gas_temp
      pfracv(3) = 1.
      PTIME(4) = tend + 2.
      et(4) = 0.
      r1t(4)= 0.
      pwc(4) = 1.
      ptemp(4) = gas_temp
      pfracv(4) = 1.
      slen = 2.*r1ss
      swid = pi*r1ss**2/slen/2.
      check4 = .true.        ! steady state run
      goto 790
c
 520  continue
c
      write(6,1450)
      read(5,*) gmass0
      write(8,1020) gmass0
C
      WRITE(6,1190)
      WRITE(6,1200)
      WRITE(6,1210)
      WRITE(6,1220)
      WRITE(6,1221)
      WRITE(6,1165)
      WRITE(6,1223)
      WRITE(6,1230)
      WRITE(6,1240)
      WRITE(6,1250)
      goto 600
C
 560  write(6,1290)
C
 600  LUNIN = 5
      WRITE(6,1330)
      READ(5,1310) NCHAR,STRING
      IF(STRING.eq.'Y' .or. string.eq.'y') goto 640
      goto 680
  640 WRITE(6,1320)
      READ(5,1310) NCHAR,STRING
      OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=560)
      LUNIN = 10
  680 CONTINUE
      IF(LUNIN .EQ. 5) WRITE(6,1260) igen
      READ(LUNIN,*) NP
      IF(LUNIN .EQ. 5) WRITE(6,1270)
C
      DO 720 I=1,NP
      READ(LUNIN,*) PTIME(I),ET(I),R1T(I)
      pwc(i) = 1.
      ptemp(i) = gas_temp
      pfracv(i) = 1.
  720 CONTINUE
      IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
      GOTO 790
C
C*** FOR A DILUTED SOURCE ...
C
 730  check4 = .false.
      write(6,1400)
      read(5,1410) dummy
      if(dummy.eq.'y' .or. dummy.eq.'Y') goto 740
      goto 750
 740  continue
      gmass0 = 0.          ! no initial cloud for a SS simulation
      write(8,1020) gmass0
      write(6,1420)
      read(5,*) ess
      write(6,1430)
      read(5,*) r1ss
 741  WRITE(6,1470)
      read(5,*) PWC(1)
      if(pwc(1).le.0.D0 .or. pwc(1).gt.1.D0) goto 741
 742  WRITE(6,1480)
      read(5,*) PTEMP(1)
      if(pwc(1).le.0.D0) goto 742
      np = 4
c
```

```
      tend = 6023. ! [=] sec
c
      PTIME(1) = 0.
      et(1) = ess
      r1t(1)= r1ss
      PWC(1)= pwc(1)
      PTEMP(1)= ptemp(1)
      PFRACV(1)= 1.0
      PTIME(2) = tend
      et(2) = ess
      r1t(2)= r1ss
      PWC(2)= pwc(1)
      PTEMP(2)=ptemp(1)
      PFRACV(2)= 1.0
      PTIME(3) = tend + 1.
      et(3) = 0.
      r1t(3)= 0.
      PWC(3)= pwc(1)
      PTEMP(3)=ptemp(1)
      PFRACV(3)= 1.0
      PTIME(4) = tend + 2.
      et(4) = 0.
      r1t(4)= 0.
      PWC(4)= pwc(1)
      PTEMP(4)=ptemp(1)
      PFRACV(4)= 1.0
      slen = 2.*r1ss
      swid = pi*r1ss**2/slen/2.
      check4 = .true.        ! steady state run
      goto 790
c
 750  continue
c
      write(6,1450)
      read(5,*) gmass0
      write(8,1020) gmass0
C
      WRITE(6,1190)
      WRITE(6,2200)
      WRITE(6,2210)
      WRITE(6,2220)
      WRITE(6,2221)
      WRITE(6,1165)
      WRITE(6,2223)
      WRITE(6,2230)
      WRITE(6,2240)
      WRITE(6,2250)
      goto 760
C
 755  write(6,1290)
C
```

```
 760  LUNIN = 5
      WRITE(6,1330)
      READ(5,1310) NCHAR,STRING
      IF(STRING.eq.'Y' .or. string.eq.'y') goto 765
      goto 770
  765 WRITE(6,1320)
      READ(5,1310) NCHAR,STRING
      OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=755)
      LUNIN = 10
  770 CONTINUE
      IF(LUNIN .EQ. 5) WRITE(6,1260) igen
      READ(LUNIN,*) NP
      IF(LUNIN .EQ. 5) WRITE(6,1270)
C
      DO 780 I=1,NP
      PFRACV(I) = 1.
      READ(LUNIN,*) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I)
  780 CONTINUE
      IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
C
  790 continue
      WRITE(8,1040) NP
      DO 800 I=1,NP
  800 WRITE(8,1030) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),PFRACV(I)
C
      if(et(1).eq.0. .and. gmass0.ne.0.) check2=.true. ! HSE type spill
      write(8,*) check1,check2,again,check3,check4,check5
C
      istat = lib$date_time(tinp)
      WRITE(8,1050) TINP
c
      if(check4) write(8,1020) ess,slen,swid        ! steady state
c
C
      CLOSE(UNIT=8)
C
c
 1010 format(1x,1pg14.7)
 1020 format(3(1x,1pg14.7))
 1025 format(5(1x,1pg14.7))
 1030 format(1x,5(1pg14.7,1x),1pg14.7)
 1040 format(1x,I5)
 1050 format(a24)
 1060 format(1x,i4,1x,1pg14.7)
c
 1100 FORMAT(5X,'INPUT MODULE -- DEGADIS MODEL')
 1110 FORMAT(/,5X,'**********************************')
 1120 FORMAT(5X,'Enter Title Block -- up to 4 lines of 80',
     $' characters')
 1130 FORMAT(5X,'To stop, type "//"')
 1134 FORMAT(A80)
```

```
1135 FORMAT(A80)
1140 FORMAT(5X,'ENTER WIND PARAMETERS -- U0 (m/s), Z0 (m), ',
    $'and ZR(m)')
1142 format(5x,'U0 -- Wind velocity at reference height Z0',
    $/,5X,'ZR -- Surface Roughness')
1150 FORMAT(/,5X,'Enter the Pasquill stability class: (A,B,C,',
    $'D,E,F) <D> ',$)
1152 format(' Enter the averaging time (s) for estimating DELTA: ',$)
1159 format(/,' The values for the atmospheric parameters',
    $' are set as follows:',
    $/,' DELTA:                       ',F12.4,
    $/,' BETA:                        ',F12.4,
    $/,' Monin-Obukhov length:        ',F12.4,' m',
    $/,' Sigma X Coefficient:         ',F12.4,
    $/,' Sigma X Power:               ',F12.4,
    $/,' Sigma X Minimum Distance: ',F12.4,' m',
    $/,' Do you wish to change any of these?',
    $/,' (No,Delta,Beta,Length,Coefficient,Power,Minimum) <N> ',$)
1160 format(/,' The values for the atmospheric parameters',
    $' are set as follows:',
    $/,' DELTA:                       ',F12.4,
    $/,' BETA:                        ',F12.4,
    $/,' Monin-Obukhov length:        infinite',
    $/,' Sigma X Coefficient:         ',F12.4,
    $/,' Sigma X Power:               ',F12.4,
    $/,' Sigma X Minimum Distance: ',F12.4,' m',
    $/,' Do you wish to change any of these?',
    $/,' (No,Delta,Beta,Length,Coefficient,Power,Minimum) <N> ',$)
1161 FORMAT(/,5X,'The density is determined as a function of con',
    $'centration')
1162 FORMAT(5X,'by a listing of ordered triples supplied by the ',
    $'user.')
1163 FORMAT(5X,'Use the following form:')
1164 FORMAT(/,5X,5X,'first point',6X,'-- pure air  y=0.0,Cc=0.,',
     1      'RHOG=RHOA=',F7.5' kg/m**3')
1165 FORMAT(3(15X,'.',/))
1166 FORMAT(5X,5X,'last point',7x,'-- pure gas  y=1.0,Cc=RHOE,',
     1      'RHOG=RHOE')
1170 FORMAT(/,5X,'ENTER THE NUMBER OF DATA TRIPLES (max=',i2,')',
    $' FOR THE DENSITY FUNCTION: ',$)
1180 FORMAT(/,5X,'Enter Mole frac, Cc (kg/m**3), then RHOG ',
     1      '(kg/m**3) by triples')
c
c
1190 FORMAT(/,5X,10X,'Source Description')
1200 FORMAT(1X,/,5X,'The description of the primary source mass')
1210 FORMAT(5X,'evolution rate E and radius R1 for a transient',/,
    $5x,'release is input by ordered triples as follows:')
1220 FORMAT(/,5X,3X,'first point',8X,
    $'-- t=0, E(t=0), R1(t=0) (initial, ',
    $'nonzero values)')
```

```
1221 FORMAT(5X,3X,'second point',7X,
    $'-- t=t1, E(t=t1), R1(t=t1)')
1223 FORMAT(5X,3X,'last nonzero point -- ',
    $'t=TEND, E(t=TEND), R1(t=TEND)')
1230 FORMAT(5X,3X,'next to last point -- t=TEND+1., E=0., R1=0.')
1240 FORMAT(5X,3X,'last point         -- t=TEND+2., E=0., R1=0.')
1250 FORMAT(/,5X,'Note! the final time (TEND) is the last time ',
    $'when E and R1 are non-zero.',/)
1260 FORMAT(/,5X,'Enter the number of triples (max= ',i2,')',
    $' starting with t=0. and ending',/,5x,'with t=TEND+2.',
    $' for the source description: ',$)
1270 FORMAT(/,5X,'Enter TIME (sec), EVOLUTION RATE (kg/s), ',
    $'and POOL RADIUS (m)')
1280 FORMAT(/,5X,'The suggested LOWEST CONCENTRATION OF INTEREST ',
    $'(gas_lfl/2.)',/,5x,' is ',1pg13.5,
    $' kg/m**3.  Enter the desired value: ',$)
1290 format(/,' This file was not found.')
1300 FORMAT(/,' Do you have an input file for the Density ',
    $'function? [y or N] ',$)
1310 FORMAT(Q,A20)
1320 FORMAT(' Enter the file name: [DIR]FILE_NAME.EXT ',$)
1330 FORMAT(' Do you have an input file for the Source ',
    $'Description? [y or N] ',$)
1340 format(/,' Air density corrected to ',1pg13.5' kg/m**3',/)
1341 format(/,' Contaminant density corrected to ',1pg13.5' kg/m**3',/)
c
c
1400 format(//,' Is this a Steady state simulation? <y or N> ',$)
1410 format(a4)
1415 format(a3)
1420 format(/,' Enter the desired evolution rate [=] kg/sec : ',$)
1430 format(' Enter the desired source radius [=] m : ',$)
1440 format(/,' Specification of source parameters.',/)
1450 format(//,' Enter the initial mass of pure gas',
    $' over the source. (kg)',/,' (Positive or zero): ',$)
1460 format(/,' Is this a release of pure (P) or diluted (d) material',
    $       ' specified above? <P or d> ',$)
1470 FORMAT(' Enter the desired primary source contaminant mass ',
    $       'fraction: ',$)
1480 FORMAT(' Enter the desired primary source temperature [=] K : ')
c
c
1500 format(/,' Enter the ambient temperature(C) and pressure',
    1     '(atm): ',$)
1510 format(/,' Enter the code name of the diffusing species: ',$)
1520 format(/,' The characteristics for the gas are set as follows:',/,
    $' Molecular weight:                                ',f7.2,/,
    $' Storage temperature [K]:                         ',1pg13.5,/,
    $' Density at storage temperature, PAMB  [kg/m**3]: ',1pg13.5,/,
    $' Mean Heat capacity constant                      ',1pg13.5,/,
    $' Mean Heat capacity power                         ',1pg13.5,/,
     $' Upper Flammability Limit [mole frac]             ',1pg13.5,/,
     $' Lower Flammability Limit [mole frac]             ',1pg13.5,/,
     $' Height of Flammability Limit [m]                 ',1pg13.5,/,
     $' Do you wish to change any of these? ',
     $'(No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z)',
     $' <N> ',$)
 1530 format(' Enter the desired Storage Temperature: ',$)
 1535 format(' Enter the desired Density at Storage ',
     1     'Temperature and',' ambient pressure: ',$)
 1550 format(' Enter the desired Molecular Weight: ',$)
 1570 format(' Enter the desired Mean Heat Capacity constant: ',$)
 1571 format(' Enter the desired Mean Heat Capacity power: ',$)
 1572 format(' Enter the desired Upper Flammability Limit: ',$)
 1573 format(' Enter the desired Lower Flammability Limit: ',$)
 1574 format(' Enter the desired Height for the flammable limit calcula',
     1     'tions: ',$)
 1580 format(/,' The ambient humidity can be entered as Relative ',
     1     'or Absolute.',/,' Enter either R or A <R or a>: ',$)
 1585 format(' Enter the absolute humidity (kg water/kg BDA): ',$)
 1586 format(' This is a relative humidity of ',1pg13.5,' %')
 1587 format(' Enter the relative humidity (%): ',$)
 1588 format(/,' Ambient Air density is ',1pg13.5,' kg/m**3')
c
c
 1600 format(' Enter the desired DELTA: ',$)
 1620 format(' Enter the desired BETA: ',$)
 1660 format(' Note: For infinity, ML = 0.0',/,
     $      ' Enter the desired Monin-Obukhov length: (m) ',$)
 1670 format(' Enter the desired Sigma X Coefficient: ',$)
 1680 format(' Enter the desired Sigma X Power: ',$)
 1690 format(' Enter the desired Sigma X Minimum distance: (m) ',$)
c
 2000 format(/,' Is this an Isothermal spill? <y or N> ',$)
 2020 format(/,' Is heat transfer to be included in the',
     1      ' calculations <y or N> ',$)
 2030 format(' Enter the surface temperature [=] K : ',$)
 2040 format(' Do you want to use the built in correlation,',
     1      ' the LLNL correlation, or',/,' enter',
     1      ' a particular value?',/,
     1      ' (Corr,LLNLcorr,Value)  <C> ',$)
 2043 format(/,' The form of the correlation is:',/,
     1     6x,'Q = (Vh * rho * cp) * area * (tsurf-temp)',//,
     1      ' with Vh = ',1pg13.5,' m/s.',//,
     1      ' Do you wish to change the value of Vh? (y or N): ',$)
 2045 format(' Do you want to use the built in correlation or enter',
     1      ' a particular value?',/,' <C or v> ',$)
 2050 format(' Enter the HT coefficient value [=] J/m**2/s/K : ',$)
 2100 format(/,' Is water transfer to be included in the',
     1      ' source <y or N> ',$)
 2120 format(' Enter the WT coefficient value [=] kg/m**2/s : ',$)
C
```

```
2200 FORMAT(1X,/,5X,'The description of the primary source contam-')
2210 FORMAT(5X,'inant mass rate E, radius R1, contaminant mass',/,
    $5x,'fraction Wc,s, and temperature Ts for a transient',/,
    $5x,'release is input as a function of time as follows:')
2220 FORMAT(/,5X,3X,'first point',8X,
    $'-- t=0., E(t=0), R1(t=0), Wc,s(t=0), Ts(t=0)',/,
    $        5x,3x,18x,10x,'(initial, ',
    $'nonzero values)')
2221 FORMAT(5X,3X,'second point',7X,
    $'-- t=t1, E(t=t1), R1(t=t1), Wc,s(t=t1), Ts(t=t1)')
2223 FORMAT(5X,3X,'last nonzero point -- ',
    $'t=TEND, E(t=TEND), R1(t=TEND), Wc,s(t=TEND), Ts(t=TEND)')
2230 FORMAT(5X,3X,'next to last point -- t=TEND+1., E=0., R1=0.,',
    $        ' Wc,s=1., Ts=Tamb')
2240 FORMAT(5X,3X,'last point         -- t=TEND+2., E=0., R1=0.,',
    $        ' Wc,s=1., Ts=Tamb')
2250 FORMAT(/,5X,'Note: the final time (TEND) is the last time ',
    $'when E and R1 are non-zero.',/)
2260 FORMAT(/,5X,'Enter the number of times (max= ',i2,')',
    $' starting with t=0. and ending',/,5x,'with t=TEND+2.',
    $' for the source description: ',$)
2270 FORMAT(/,5X,'Enter TIME (sec), CONTAMINANT MASS RATE (kg C/s), ',
    $'POOL RADIUS (m)',/,10X,'CONTAMINANT MASS FRACTION (kg C',
    $        '/kg mix), and TEMPERATURE (K)')
2280 FORMAT(/,5X,'The suggested LOWEST CONCENTRATION OF INTEREST ',
    $'(gas_lfl/2.)',/,5x,' is ',1pg13.5,
    $' kg/m**3.  Enter the desired value: ',$)
c
      RETURN
      END
$$$$
```

```
C..............................................................
C
C    SUBROUTINE FOR SOURCE EVALUATION WHEN NO GAS BLANKET
C       IS PRESENT.
C
      SUBROUTINE NOBL(timeout, reflag)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/ERROR/ STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/phlag/ check1,check2,again,check3,check4,check5
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
C
      REAL*8 ML,K
c
      LOGICAL REV
      logical check1,check2,again,check3,check4,check5
      logical reflag
      DATA REV/.TRUE./
      REAL*8 L
c
      data h/0./,Ri/0./
      data delt_min/0.5/
C
      DELTAT = (TEND - TSC1)/FLOAT(NOBLPT)
      if(deltat .lt. delt_min) then
            noblpt = int((tend-tsc1)/delt_min) +1
            deltat = (tend-tsc1)/float(noblpt)
            endif
C
      TO = TSC1
      IF(DELTAT .LT. 2.) GO TO 100
C
      WRITE(lunlog,1100)
      WRITE(lunlog,*) DELTAT
 1100 FORMAT(5X,'TIME INCREMENT USED ON LAST PORTION OF SOURCE CALC')
C
```

```
  100 CONTINUE
C
C     ESTABLISH LOOP TO FINISH SOURCE
C
      DO 110 I = 1,NOBLPT
C
      TIME = TO + FLOAT(I)*DELTAT
      IF(I .EQ. NOBLPT) TIME = TEND
      L = 2.0*AFGEN2(PTIME,R1T,TIME,'R1T-BL')
      erate = AFGEN2(PTIME,ET,TIME,'ET-BL')
      flux = EraTe/(pi*L*L/4.)
      PWCP = AFGEN2(PTIME,PWC,TIME,'ET-BL')
      PWAP = (1.D0 - PWCP)/(1.D0 + HUMID)
      HPRIM = AFGEN2(PTIME,PENTH,TIME,'ET-BL')
      CALL SETDEN(PWCP, PWAP, HPRIM)
      RHOP = AFGEN2(PTIME,PRHO,TIME,'RH-BL')
      CCP = PWCP * RHOP
C
      astar = CCP * k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L)
      if(abs(flux/astar) .gt. ernobl .and. reflag) then
            check3 = .true.
            timeout = time
            return
            end if
C
      call szf(flux,L,PWCP,sz,cclay,wclay,rholay)
      cc = cclay*dellay
      if(cc.gt.ccp) cc=ccp
C
      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
C
      IF(Erate .LT. EMAX) GO TO 220
      EMAX = Erate
      RM = AFGEN2(PTIME,R1T,TIME,'R1T-BL')
      SZM = SZ
  220 CONTINUE
      RLIST = AFGEN2(PTIME,R1T,TIME,'R1T-BL')
      RMAX = dMAX1(RMAX,RLIST)
C
      WRITE(9,2000) TIME,RLIST,h,flux,SZ,yc,ya,rho,Ri,wc,wa,enthalpy,temp
C
      if(i.eq.3  .and.  check4) goto 500 ! steady
C
  110 CONTINUE
      RETURN
C
C
  500 continue               ! steady state completion
      outcc = cc
      swcl = wc
      swal = wa
      senl = enthalpy
      srhl = rho
      outsz = sz
      outl  = 2. * rlist
      outb  = pi * rlist**2 /outl/2.
      return
 2000 format(1pg16.9,1x,1pg16.9,<iout_src-2>(1x,1pg13.6))
      END
####
```

```
C.......................................................
C
C     SUBROUTINES OB AND OBOUT ARE USED IN THE OBSERVER INTEGRATIONS
C          OVER THE SOURCE.
C
      SUBROUTINE OB(time,Y,D,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      include 'sys$degadis:DEGADIS2.dec'
c
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
     $/phicom/ iphifl,dellay
c
      REAL*8 K,ML
      logical flag
C
      DIMENSION Y(1),D(1),PRMT(1)
      INTEGER HWIDTH,Mrate,Crate,BDArate,Hrate
      DATA HWIDTH/1/,Mrate/2/,Crate/3/,BDArate/4/,Hrate/5/
C
C*** PASS TO IN PRMT(6)
C
      flag = isofl.eq. 1 .or. ihtfl.eq. 0
c
      TOL = PRMT(6)
      xup = prmt(7)
      XI = XIT(TIME,T01)
      RG = AFGEN(RADG,TIME,'RADG')
      RLEN = PRMT(13)
C
      BIPR = 0.
      D(HWIDTH)= 0.
      D(Crate) = 0.
      D(Mrate) = 0.
      D(BDArate)= 0.
      D(Hrate) = 0.

c
c*** We must deal with the case when the observer sees no gas, but the
c*** integration is not over yet.  Since the derivatives have been set
c*** to zero above, no work on the derivatives is needed, but the values
c*** in PRMT(14-18) must be preserved.  Since these will be switched
c*** in OBOUT, switch them now so the old values will be retained.
```

```
c
      IF(ABS(XI) .GE. RG) then    ! use the last values
            PRMT(8)  = prmt(14)   ! cclay
            PRMT(9)  = prmt(15)   ! wclay
            PRMT(10) = prmt(16)   ! walay
            PRMT(11) = prmt(17)   ! enthlay
            PRMT(12) = prmt(18)   ! rholay
            RETURN
            endif
      BIPR = sqrt(RG*RG - XI*XI)
C
      UI = UIT(TIME,T01)
C
      Q    = AFGEN(QSTR,TIME,'QSTR')
      wc   = AFGEN(srcwc,time,'srcwc')
      wa   = AFGEN(srcwa,time,'srcwa')
      enth = AFGEN(srcenth,time,'srcenth')
c
      wclay   = Y(Crate)/Y(Mrate)
      walay   = Y(BDArate)/Y(Mrate)
      if(.not.flag) enthlay = Y(Hrate)/Y(Mrate)
c
      call tprop(1,wclay,walay,enthlay,yc,ya,wm,temp,rholay,cp)
      cclay = wclay*rholay
c
      prmt(8) = cclay
      prmt(9) = wclay
      prmt(10)= walay
      prmt(11)= enthlay
      prmt(12)= rholay
c
      cc  = cclay*dellay
      rho = dellay*(rholay-rhoa) + rhoa
c
      szob = 0.01
      arg = Q*(xi-xup)/cc/(u0*z0/alpha1)
      if(xi.gt. xup .and. arg.gt.0.)
      1 szob = arg**(1./alpha1) * z0
c
      HEFF = GAMMAF/ALPHA1* SZOB
      RISTR=RIF(RHO,HEFF)
      PHI = PHIF(RISTR,0.)
      welay = dellay * K*USTAR* ALPHA1/PHI
C
      D(HWIDTH)= UI * BIPR / RLEN
      D(Crate) = D(HWIDTH)*RLEN * Q
      D(Mrate) = (Q/wc + rhoa*welay) * D(HWIDTH)*RLEN
      D(BDArate)= (Q*wa/wc + rhoa*welay/(1.+humid)) * D(HWIDTH)*RLEN
      if(flag) return
      D(Hrate) = Q * enth/wc * D(HWIDTH)*RLEN
C
```

```
 1000 FORMAT(' ?OB? -- Value of XI ',1pG13.4,'; Value of RG ',
     $  1pG13.4)
      RETURN
      END
C
C
      SUBROUTINE OBOUT( X, Y, DERY, IHLF, NDIM, PRMT)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      DIMENSION X(1), Y(1), DERY(1), PRMT(1)
C
      PRMT(14) = prmt(8)    ! cclay
      PRMT(15) = prmt(9)    ! wclay
      PRMT(16) = prmt(10)   ! walay
      PRMT(17) = prmt(11)   ! enthlay
      PRMT(18) = prmt(12)   ! rholay
      RETURN
      END
$$$$
```

```
C.............................................................
C
C     FUNCTION PSI
C
C*** AS PER COLENBRANDER --
C
C*** THIS FUNCTION HAS BEEN DERIVED FROM BUSINGER,J.A.
C***  WORKSHOP ON MICROMETEOROLOGY, CHAPTER 2, HAUGEN,D.A. (ED.)
C***  AMERICAN METEOROLOGICAL SOCIETY.
C
      FUNCTION PSIF(Z,ML)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS1.dec'
c
      REAL*8 ML
C
      IF( ML ) 10,20,30
C
 10   A = (1.-15.*Z/ML)**.25
      PSIF = 2.*dLOG((1.+A)/2.) + dLOG((1.+A*A)/2.) - 2.*daTAN(A) +
     $ PI/2.
      RETURN
C
 20   PSIF = 0.
      RETURN
C
 30   PSIF = -4.7*Z/ML
      RETURN
      END
$$$$
```

```
C.........................................................
C
C     SUBROUTINES FOR PSEUDO-STEADY STATE INTEGRATION.
C
      SUBROUTINE PSS(DIST,Y,DERY,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      include 'sys$degadis:DEGADIS2.dec/list'
c
      parameter (zero=1.D-10, rcrit=2.D-3)
c
      COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
c
      REAL*8 K,ML

      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      DIMENSION Y(1),DERY(1),PRMT(1)
      DATA rhouh/1/,SY2/2/,BEFF/3/,dh/4/,Mhi/5/,Mlow/6/
      INTEGER rhouh,SY2,BEFF,dh, Mhi, Mlow
C
C*** PRMT I/O SETUP
C***  I     VALUE          IN/OUT
C***  ----  -----          -----
C***   6     E              IN
C***   7     Cc            OUT
C***   8     Bb            OUT
C***   9    CON DERY(BEFF)  IN
C***  10    CON DERY(SZ)    IN
C***  11    NREC(I,1)       OUT -- STARTS OUTPUT COUNTER
C***  12    DIST           OUT
c***  13
c***  14    yc             out
c***  15    rho            out
c***  16    temp           out - if recorded
c***  17    gamma          out - if recorded
c***  18
c***  19
c***  20
c***  21    sz
c***  22    sz
```

```
C
      Erate = PRMT(6)
      Bb = Y(BEFF) - SQrtPI/2.*sqrt(Y(SY2))
c
c using the last value for Sz
c
      sz0 = prmt(22)
      sz = sz0
C
C*** MATERIAL BALANCE
C
      iii = 0
 100  Cc = Erate*ALPHA1/2./U0*(Z0/SZ)**ALPHA/SZ/Y(BEFF)
      cclay = cc/dellay
      call addheat(cclay,y(dh),rholay,temlay,cp)
      prod = dmax1( Y(rhouh)/rholay/prmt(18), zero)
      sz = ( prod )**(1./alpha1) * z0
      dif = abs(sz - sz0)/(abs(sz)+abs(sz0)+zero)
      if(dif .gt. rcrit) then
            sz0 = sz
            iii=iii+1
            if(iii .gt. 20) call trap(32)
            goto 100
            endif
      prmt(20) = rholay
      prmt(21) = sz
      HEFF = GAMMAF/ALPHA1*SZ

      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
      call adiabat(0,wc,wa,yclay,ya,cclay,rholam,wml,enth,temlay)
c
      rit = 0.
      if(isofl.eq.0 .and. ihtfl.ne.0) then
            call addheat(cc,dellay*y(dh),rho,temp,cp)
            rit = rift(temp,heff)
            endif
      RISTR = RIF(RHO,HEFF)
      PHI = PHIF(RISTR,rit)
C
C*** CALCULATE DERIVATIVES
C
      DERY(BEFF) = 0.
      delrho = rho-rhoa
      IF(delrho .GT. delrhomin) DERY(BEFF) = PRMT(9)*sqrt(delrho/rhoa)
     $    *(SZ/Z0)**(.5 - ALPHA)
C
      DERY(SY2) = 8.*BETA/PI*Y(BEFF)**2 *
     $    (DELTA*SQPIO2/Y(BEFF)) ** (1./BETA)
C
c
      heigh = heff*dellay
      yw = 1.-yclay-ya
      yw = min( max( yw, 0.D0 ), 1.D0)
      call surface(temlay,heigh,rholay,wml,cp,yw,watrte,qrte)
      if(temp.ge. tsurf .or. temlay.ge. tamb) qrte = 0.
      rhouhb = rholay* prmt(18) * (sz/z0)**alpha1 * Y(beff)
      d_rhouhb = prmt(19)*y(beff)/phi
      dERY(dh) = (qrte*Y(beff)/dellay - Y(dh)*d_rhouhb)/rhouhb
      dERY(rhouh) = (d_rhouhb-Y(rhouh)*DERY(beff))/Y(beff)
c
c*** Calculate the derivative for the total mass above the UFL and LFL
c
      gamma = (rho-rhoa)/cc          ! gamma
      if(check4) then
        DERY(mlow) = 0.
        DERY(mhi)  = 0.
        if( isofl.eq.1 .or. ihtfl.eq.0 ) then
            call adiabat(2,aa,dd,gas_ufl,ee,chi ,gg,hh,pp,oo)
            call adiabat(2,aa,dd,gas_lfl,ee,clow,gg,hh,pp,oo)
        else
            call adiabat(-2,aa,dd,gas_ufl,ee,chi ,gg,hh,gamma,oo)
            call adiabat(-2,aa,dd,gas_lfl,ee,clow,gg,hh,gamma,oo)
        endif
            gamhi  = 2.D0* Cc * Bb * Sz / alpha1
      gammax = 2.D0* Cc * Sz * GAMMAF/alpha1 * (Bb+sqrtpi/2.D0*sqrt(Y(Sy2)))

        if(cc.gt.clow) then
            wlow = Dlog(cc/clow)
            gamlow = gaminc(1.D0/alpha1, wlow ) * gamhi
          DERY(mlow)= gamlow + 2.D0*clow*sqrt(Y(Sy2))*Sz/alpha1*series(wlow)
          DERY(mlow)= DMIN1( DERY(mlow), gammax )
        endif

        if(cc.gt.chi) then
            whi  = Dlog(cc/chi )
            gamhi  = gaminc(1.D0/alpha1, whi  ) * gamhi
          DERY(mhi) = gamhi  + 2.D0*chi *sqrt(Y(Sy2))*Sz/alpha1*series(whi )
          DERY(mhi) = DMIN1( DERY(mhi ), gammax )
        endif
      endif
C
C*** RETURNED VALUES
C
      PRMT(7) = Cc
      PRMT(8) = Bb
      prmt(14)= yc
      prmt(15)= rho
      prmt(16)= temp
      prmt(17)= gamma
      RETURN
      END
####
```

```
C.............................................................
C
C     SUBROUTINE PSSOUT
C
      SUBROUTINE PSSOUT(X,Y,D,IHLF,NDIM,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS2.dec'
c
      parameter (npss=9, zero=1.e-10)
c
      COMMON
     $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/STP/STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/STOPIT/TSTOP
c
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      DIMENSION Y(1),D(1),PRMT(1),BKSP(npss),OUT(npss),CURNT(npss)
C
C*** OUTPUT PARAMETERS
C
C*** FROM PSS                OUTPUT TO MODEL
C*** --------                ---------------
C***  X             DIST
C***  PRMT(7)                  Cc
C***  Y(1)          SZ
C***  Y(2)          SY2
C***  PRMT(8)                  B
C***  PRMT(13)                TO(I)
c***  prmt(14)                yc
c***  prmt(15)                rho
c***  prmt(16)                temp
c***  prmt(17)                gamma
c
      ERM = 0.
      TSL = TS(PRMT(13),X)
      prmt(22) = prmt(21)
      IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR THE OUTPUT ROUTINE
C
      RII = -100./STPP
      RI = 0.
      CURNT(1) = X
      curnt(2) = prmt(14)  ! yc
      CURNT(3) = PRMT(7)    ! cc
      curnt(4) = prmt(15)   ! rho
      curnt(5) = prmt(17)   ! gamma
      curnt(6) = prmt(16)   ! temp
      CURNT(7) = prmt(21)   ! sz
      CURNT(8) = sqrt(Y(2)) ! sy2
      CURNT(9) = PRMT(8)    ! b
      IF(prmt(8) .LE. 0.) CALL trap(16)
   90 CONTINUE
C
C*** STOP INTEGRATION WHEN THE HALF WIDTH B < 0.
C
      IF( PRMT(8) .LE. 0.) GO TO 1000
C
C*** SET THE CURRENT AND PREVIOUS RECORD
C
      DO 100 II=1,npss
  100 BKSP(II) = CURNT(II)
C
      CURNT(1) = X
      curnt(2) = prmt(14)   ! yc
      CURNT(3) = PRMT(7)    ! cc
      curnt(4) = prmt(15)   ! rho
      curnt(5) = prmt(17)   ! gamma
      curnt(6) = prmt(16)   ! temp
      CURNT(7) = prmt(21)   ! sz
      CURNT(8) = sqrt(Y(2)) ! sy2
      CURNT(9) = PRMT(8)    ! b
C
C*** STOP INTEGRATION AND GET A NEW OBSERVER WHEN Cc<CcLOW
C
      IF(PRMT(7).GT.CcLOW .OR. TSL.LT.TSTOP) GO TO 95
            if(prmt(11) .lt. 5.) then    ! guarantee 5 records
             erm = odlp      ! force output
             goto 95
             endif
      AGAIN = .TRUE.
      TSTOP = TSL
      GO TO 1000
   95 CONTINUE
C
c*** Check the error criteria's
c
      RI = RI + 1.
      II = 2              ! skip DIST,YC; 1 and 2
  110 II = II + 1
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
      ERM = dMAX1(ER1,ER2,ERM)
      IF(II .EQ. 3) II = II + 1   ! skip RHO;4
      IF(II .EQ. 7) II = II + 1   ! skip SY;8
      IF(II .EQ. 5) II = II + 1   ! skip TEMP;6
      IF(II .LT. npss) GO TO 110
C
C*** RECORD POINT IF ODLP IS EXCEEDED OR 80 METERS SINCE LAST RECORD
C*** RECORD FIRST POINT
C
      DX = CURNT(1) - OUT(1)
      IF( RI.NE.1. .AND. ERM.LT.ODLP .AND. DX.LE.ODLLP) RETURN
C
C*** IF THE NEXT INTEGRATION AFTER A POINT IS RECORDED VIOLATES THE
C*** ERROR BOUND, THE CURRENT POINT MUST BE RECORDED. OTHERWISE, THE
C*** LAST POINT TO SATISFY THE ERROR LIMITS IS RECORDED.
C
      DO 120 II=1,npss
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      WRITE(9,*) (OUT(II),II=1,npss)
      RETURN
C
 1000 CONTINUE
C
C*** STOP INTEGRATION
C
      PRMT(12) = X
C
      IF(CURNT(1) .EQ. OUT(1)) GO TO 130
C
      PRMT(11) = PRMT(11) + 1.
      WRITE(9,*) (CURNT(II),II=1,npss)
C
  130 CONTINUE
      PRMT(5) = 1.
      RETURN
      END
####
```

```
C.............................................................
C
C     SUBROUTINE PSSOUT
C
      SUBROUTINE PSSOUT(X,Y,DERY,IHLF,NDIM,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS2.dec/list'
c
      parameter (npss=9, zero=1.e-10)
c
      COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_fl/ cflag,clfl,cufl
     $/ALP/ALPHA,alpha1
C
      logical cflag
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      REAL*8 ML,K
C
      DIMENSION Y(1),DERY(1),PRMT(1)
      dimension BKSP(npss),OUT(npss),CURNT(npss)
C
C*** OUTPUT PARAMETERS
C
C*** FROM PSS                 OUTPUT TO MODEL
C*** --------                 ---------------
C***  X             DIST
C***  PRMT(7)                Cc
C***  Y(1)          SZ
C***  Y(2)          SY2
C***  PRMT(8)                B
C
      ERM = 0.
      prmt(22) = prmt(21)
C
      IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR THE OUTPUT ROUTINE
C
      RII = -100./STPP
      RI = 0.
      CURNT(1) = X
      CURNT(2) = PRMT(14)  ! yc
      CURNT(3) = prmt(7)   ! cc
      CURNT(4) = prmt(15)  ! rho
      CURNT(5) = PRMT(17)  ! gamma
      curnt(6) = prmt(16)  ! temp
      curnt(7) = prmt(8)   ! b
      curnt(8) = prmt(21)  ! sz
      curnt(9) = sqrt(Y(2)) ! sy2
C
   90 CONTINUE
C
C*** STOP INTEGRATION WHEN THE HALF WIDTH B < 0.
C
      IF( PRMT(8) .LE. 0.) GO TO 1000
C
C*** STOP INTEGRATION when Cc<CcLOW
C
      IF(PRMT(7).GT.CcLOW) GO TO 95
             if(prmt(11) .lt. 5.) then    ! force output
              erm = odlp
              goto 95
              endif
      AGAIN = .TRUE.
      GO TO 1000
   95 CONTINUE
C
C*** SET THE CURRENT AND PREVIOUS RECORD
C
      DO 100 II=1,npss
  100 BKSP(II) = CURNT(II)
C
      CURNT(1) = X
      CURNT(2) = PRMT(14)  ! yc
      CURNT(3) = prmt(7)   ! cc
      CURNT(4) = prmt(15)  ! rho
      CURNT(5) = PRMT(17)  ! gamma
      curnt(6) = prmt(16)  ! temp
      curnt(7) = prmt(8)   ! b
      curnt(8) = prmt(21)  ! sz
      curnt(9) = sqrt(Y(2)) ! sy2
C
      RI = RI + 1.
      II = 1
  110 II = II + 1
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
      ERM = dMAX1(ER1,ER2,ERM)
      IF(II .EQ. 3) II = 6 ! skip density,gamma,temp
      IF(II .LT. npss-1) GO TO 110  ! skip sy
C
C*** RECORD POINT IF ODLP IS EXCEEDED OR 80 METERS SINCE LAST RECORD
C*** RECORD FIRST POINT
C
      DX = CURNT(1) - OUT(1)
      IF( RI.NE.1. .AND. ERM.LT.ODLP .AND. DX.LE.ODLLP) RETURN
C
C*** IF THE NEXT INTEGRATION AFTER A POINT IS RECORDED VIOLATES THE
C*** ERROR BOUND, THE CURRENT POINT MUST BE RECORDED. OTHERWISE, THE
C*** LAST POINT TO SATISFY THE ERROR LIMITS IS RECORDED.
C
      DO 120 II=1,npss
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      call ssout(out)
      RETURN
C
 1000 CONTINUE
C
C*** STOP INTEGRATION
C
      PRMT(12) = X
C
      IF(RI .EQ. 0.) CALL trap(16)
C
      IF(CURNT(1) .EQ. OUT(1)) GO TO 130
C
      PRMT(11) = PRMT(11) + 1.
      call ssout(out)
C
  130 CONTINUE
      PRMT(5) = 1.
C
      RETURN
      END
$$$$
```

```
C.........................................................
C
C     RICHARDSON NUMBER (RI*)
C
      FUNCTION RIF(RHOG,HEFF)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      COMMON
     $ /PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C
      REAL*8 ML,K
C
      RIF = G*(RHOG-RHOA)/RHOA*HEFF/USTAR/USTAR
C
      RETURN
      END
c
C.........................................................
C
C     RICHARDSON NUMBER (RIt)
C
      FUNCTION RIFt(temp,HEFF)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      COMMON
     $ /PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/alp/ alpha,alpha1
C
      REAL*8 ML,K
C
      wind = u0*(heff/z0)**alpha
      RIFt = dmax1(G*(tsurf-temp)/temp*HEFF/USTAR/wind,0.D0)
C
      RETURN
      END
C
C.................................................................
C
C     PHI FUNCTION
C
      FUNCTION PHIF(RI,rit)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
```

```
      common /phicom/ iphifl,dellay
C
      phif= 0.
      goto(10,1000,2000,3000,9000),iphifl
      goto 9000
c
 10   IF(RI) 100,200,300
C
 100  PHIF = 0.74/(1. + 0.65*ABS(RI)**.6)
      RETURN
C
 200  PHIF = 0.74
      RETURN
C
 300  PHIF = 0.74 + 0.25*(RI)**0.7 + 1.2E-7*RI*RI*RI
      RETURN
c
c
 1000 IF(RI) 1100,1200,1300
C
 1100 PHIF = 0.88/(1. + 0.65*ABS(RI)**.6)
      RETURN
C
 1200 PHIF = 0.88
      RETURN
C
 1300 PHIF = 0.88 + 9.9e-2*(RI)**1.04 + 1.4E-25*RI**5.7
      RETURN
c
c
c
 2000 corr1 = 0.25* rit**.666666 + 1.
      corr = sqrt(corr1)
      riw = ri/corr1
      IF(RI) 2100,2200,2300
C
 2100 PHIF = 0.88/(1. + 0.65*ABS(RIw)**.6)/corr
      RETURN
C
 2200 PHIF = 0.88/corr
      RETURN
C
 2300 PHIF = (0.88 + 9.9e-2*(RIw)**1.04 + 1.4E-25*RIw**5.7)/corr
      RETURN
c
c
 3000 corr1 = 0.25* rit**.666666 + 1.
      corr = sqrt(corr1)
      riw = ri/corr1
      IF(RI) 3100,3200,3300
C
```

```
 3100 PHIF = 0.88/corr
      RETURN
C
 3200 PHIF = 0.88/corr
      RETURN
C
 3300 PHIF = (0.88 + 9.9e-2*(RIw)**1.04 + 1.4E-25*RIw**5.7)/corr
      RETURN
c
c
 9000 call trap(29)
      return
      END
c
c
c.........................................................
c
c
      function phihat(rho,fetch)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
      common
     $/parm/ u0,z0,zr,ml,ustar,k,g,rhoe,rhoa,delta,beta,gammaf,cclow
     $/alp/ alpha,alpha1
     $/phicom/ iphifl,dellay
c
      real*8 k,ml
c
      data phic/3.1/
c
      if(rho .le. rhoa) then
            phihat = 0.88
            return
            endif
c
      pow = 1./alpha1
      p1 = 1.04/alpha1
      Ci = g*(rho-rhoa)/rhoa*z0/ustar**2*gammaf/alpha1
      Ci =Ci* (k*ustar*alpha1**2 /u0/z0/ phic*dellay/(dellay-1.)) ** pow
c
      Cip = 0.099*Ci**1.04
c
      phihat = dlog((.88+ Cip*fetch**p1)/.88)/Cip/fetch
      phihat = 1./phihat
c
      return
      end
####
```

```
C     ...................................................
C
C     SUBROUTINE RKGST
C
C.........................................................
C
C     This routine was originally supplied by Digital Equipment
C     Corporation as part of the Scientific Subroutine Package
C     available for RT-11 as part of the Fortran Enhancement
C     Package.  It was upgraded for use as the integration
C     routine in this package.
C
C.........................................................
C
C     PURPOSE
C           TO SOLVE A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL
C           EQUATIONS WITH GIVEN INITIAL VALUES.
C
C     USAGE
C           CALL RKGST (PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)
C           PARAMETERS FCT AND OUTP REQUIRE AN EXTERNAL STATEMENT.
C
C     DESCRIPTION OF PARAMETERS
C
C     PRMT  AN INPUT AND OUTPUT VECTOR WITH DIMENSION GREATER
C           OR EQUAL TO 5, WHICH SPECIFIES THE PARAMETERS OF
C           THE INTERVAL AND OF ACCURACY AND WHICH SERVES FOR
C           COMMUNICATION BETWEEN SUBROUTINES OUTP AND FCT
C           (FURNISHED BY THE USER) AND SUBROUTINE RKGST.
C           EXCEPT PRMT(5) THE COMPONENTS ARE NOT DESTROYED
C           BY SUBROUTINE RKGST AND THEY ARE:
C     PRMT(1)       LOWER BOUND OF THE INTERVAL (INPUT),
C     PRMT(2)       UPPER BOUND OF THE INTERVAL (INPUT),
C     PRMT(3)       INITIAL INCREMENT OF THE INDEPENDENT VARIABLE
C           (INPUT),
C     PRMT(4)       UPPER ERROR BOUND (INPUT). IF RELATIVE ERROR IS
C           GREATER THAN PRMT(4), INCREMENT GETS HALVED.
C           IF RELATIVE ERROR LESS THAN PRMT(4)*EXPAND,
C           INCREMENT GETS DOUBLED.
C           THE USER MAY CHANGE PRMT(4) BY MEANS OF HIS
C           OUTPUT SUBROUTINE.
C     PRMT(5)       MAXIMUM STEP SIZE ORDER OF MAGNITUDE (INPUT).
C           SUBROUTINE RKGST INITIALIZES
C           PRMT(5)=0. IF THE USER WANTS TO TERMINATE
C           SUBROUTINE RKGST AT ANY OUTPUT POINT, HE HAS TO
C           CHANGE PRMT(5) TO NON-ZERO BY MEANS OF SUBROUTINE
C           OUTP. FURTHER COMPONENTS OF VECTOR PRMT ARE
C           FEASIBLE IF ITS DIMENSION IS DEFINED GREATER
C           THAN 5. HOWEVER SUBROUTINE RKGST DOES NOT REQUIRE
C           AND CHANGE THEM. NEVERTHELESS THEY MAY BE USEFUL
C           FOR HANDING RESULT VALUES TO THE MAIN PROGRAM
```

```
C              (CALLING RKGST) WHICH ARE OBTAINED BY SPECIAL
C              MANIPULATIONS WITH OUTPUT DATA IN SUBROUTINE OUTP.
C        Y     INPUT VECTOR OF INITIAL VALUES.  (DESTROYED)
C              LATER, Y IS THE RESULTING VECTOR OF DEPENDENT
C              VARIABLES COMPUTED AT INTERMEDIATE POINTS X.
C        DERY  INPUT VECTOR OF ERROR WEIGHTS.  (DESTROYED)
C              ERROR WEIGHTS ARE CENTERED AT ONE. IF ONE PARA-
C              METER NEEDS A TIGHTER ERROR CRITERIA,THE WEIGHT IS
C              GREATER THAN ONE. IF A PARAMETER NEED NOT BE DETER-
C              MINED SO PRECISELY,THE WEIGHT SHOULD BE LESS
C              THAN ONE.IN OTHER WORDS,
C               ERROR CRITERIA(I) = PRMT(4) / WEIGHT(I)
C              WHERE I IS THE SUBSCRIPT OF A DEPENDENT VARIABLE.
C              LATER, DERY IS THE VECTOR OF DERIVATIVES, WHICH
C              BELONG TO FUNCTION VALUES Y AT A POINT X.
C        NDIM  AN INPUT VALUE, WHICH SPECIFIES THE NUMBER OF
C              EQUATIONS IN THE SYSTEM.
C        IHLF  AN OUTPUT VALUE, WHICH SPECIFIES THE NUMBER OF
C              BISECTIONS OF THE INITIAL INCREMENT. IF IHLF BE-
C              COMES GREATER THAN 10, SUBROUTINE RKGST RETURNS THE
C              ERROR MESSAGE IHLF=11 INTO MAIN PROGRAM. ERROR
C              MESSAGE IHLF=12 OR IHLF=13 APPEARS IN CASE
C              PRMT(3)=0 OR IN CASE SIGN(PRMT(3)).NE.SIGN(PRMT(2)-
C              PRMT(1)) RESPECTIVELY.
C        FCT   THE NAME OF AN EXTERNAL SUBROUTINE USED. THIS
C              SUBROUTINE COMPUTES THE RIGHT HAND SIDES DERY OF
C              THE SYSTEM TO GIVEN VALUES X AND Y. ITS PARAMETER
C              LIST MUST BE X,Y,DERY,PRMT. SUBROUTINE FCT SHOULD
C              NOT DESTROY X AND Y.
C        OUTP  THE NAME OF AN EXTERNAL OUTPUT SUBROUTINE USED.
C              ITS PARAMETER LIST MUST BE X,Y,DERY,IHLF,NDIM,PRMT.
C              NONE OF THESE PARAMETERS (EXCEPT, IF NECESSARY,
C              PRMT(4),PRMT(5),...) SHOULD BE CHANGED BY
C              SUBROUTINE OUTP. IF PRMT(5) IS CHANGED TO NON-ZERO,
C              SUBROUTINE RKGST IS TERMINATED.
C        AUX   AN AUXILIARY STORAGE ARRAY WITH 8 ROWS AND NDIM
C              COLUMNS.
C
C        REMARKS
C
C        THE PROCEDURE TERMINATES AND RETURNS TO CALLING PROGRAM, IF
C        (1) MORE THAN 10 BISECTIONS OF THE INITIAL INCREMENT ARE
C            NECESSARY TO GET SATISFACTORY ACCURACY (ERROR MESSAGE
C            IHLF=11),
C        (2) INITIAL INCREMENT IS EQUAL TO 0 OR HAS WRONG SIGN
C            (ERROR MESSAGES IHLF=12 OR IHLF=13),
C        (3) THE WHOLE INTEGRATION INTERVAL IS WORKED THROUGH,
C        (4) SUBROUTINE OUTP HAS CHANGED PRMT(5) TO NON-ZERO.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        THE EXTERNAL SUBROUTINES FCT(X,Y,DERY,PRMT) AND
```

```
C     OUTP(X,Y,DERY,IHLF,NDIM,PRMT) MUST BE FURNISHED BY THE USER.
C
C     METHOD
C     EVALUATION IS DONE BY MEANS OF FOURTH ORDER RUNGE-KUTTA
C     FORMULAE IN THE MODIFICATION DUE TO GILL. ACCURACY IS
C     TESTED COMPARING THE RESULTS OF THE PROCEDURE WITH SINGLE
C     AND DOUBLE INCREMENT.
C     SUBROUTINE RKGST AUTOMATICALLY ADJUSTS THE INCREMENT DURING
C     THE WHOLE COMPUTATION BY HALVING OR DOUBLING. IF MORE THAN
C     10 BISECTIONS OF THE INCREMENT ARE NECESSARY TO GET
C     SATISFACTORY ACCURACY, THE SUBROUTINE RETURNS WITH
C     ERROR MESSAGE IHLF=11 INTO MAIN PROGRAM.
C     TO GET FULL FLEXIBILITY IN OUTPUT, AN OUTPUT SUBROUTINE
C     MUST BE FURNISHED BY THE USER.
C     FOR REFERENCE, SEE
C     RALSTON/WILF, MATHEMATICAL METHODS FOR DIGITAL COMPUTERS,
C     WILEY, NEW YORK/LONDON, 1960, PP.110-120.
C
C     SOME NOTES ON THE PROGRAM/RALSTON AND WILF
C
C     AUX
C     ---
C
C     AUX(1,I) -- CURRENT VALUE OF Y
C     AUX(2,I) -- CURRENT VALUE OF Y'
C     AUX(3,I) -- LAST GOOD VALUES OF Q
C     AUX(4,I) -- Y AFTER ONE RK STEP H
C     AUX(5,I) -- Y AFTER ONE OR TWO RK STEPS OF H/2.
C     AUX(6,I) -- CURRENT VALUES OF Q
C     AUX(7,I) -- Y' AFTER ONE OR TWO RK STEPS OF H/2.
C     AUX(8,I) -- 2/15 * WEIGHTS
C
C     A(4),B(4),C(4)
C     --------------
C
C     Y  = Y    + A  *(K   -  B *Q   )
C      i    i-1    i    i    i  i-1
C
C     Q  = Q    + 3*(A *(K  - B *Q     ) - C *K
C      i    i-1       i   i    i  i-1       i  i
C
C     FOR VALUES OF I BETWEEN 1 AND 4, AND FOR VALUES OF K AS FOLLOWS
C
C     K  = H * F(X ,Y )
C      1          0  0
C
C     K  = H * F(X +H/2,Y )
C      2          0      1
C
C     K  = H * F(X +H/2,Y )
C      3          0      2
```

```
C
C     K  = H * F(X +H,Y )
C      4         0   3
C
C     RELATIVE ERROR
C     -------- -----
C
C     AS PER RICHARDSON QUOTED IN RALSTON/WILF (P117),
C
C     ABS ERROR = WEIGHT/15*ABS(Y2 - Y1)
C
C     THEN, RELATIVE ERROR
C
C     REL ERROR = WEIGHT*2/15*ABS(Y2 - Y1)/SUM
c
c     where SUM = ABS(Y2 + Y1)
C
C     The solution tries to use SUM=abs(y2+y1) first. If this is zero,
C     then SUM=.25*ABS(Y1) is used since y1 and y2 must be oposite in
C     sign with equal magnitude. If this
C     quantity is zero as well, the values y2 and y1 both must be zero;
C     therefore, the difference is also zero which satisfies the
C     error criteria.
C
C     ...........................................................
C
      SUBROUTINE RKGST(PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
C
      PARAMETER (EXPAND=0.05)
      DATA ERRSET/1./        ! dummy value
C
      DIMENSION Y(1),DERY(1),AUX(8,NDIM),A(4),B(4),C(4),PRMT(1)
C
      DO 10 I=1,NDIM
   10 AUX(8,I)=.13333333333333333D0 * DERY(I)
      X=PRMT(1)
      XEND=PRMT(2)
      H=PRMT(3)
      IHLFMX = INT(dLOG(ABS(PRMT(5)/PRMT(3)))/.6931472 + .5)
      PRMT(5)=0.
      CALL FCT(X,Y,DERY,PRMT)
C
C*** ERROR TEST
C
      IF(H*(XEND-X))380,370,20
C
C*** PREPARATIONS FOR RUNGE-KUTTA METHOD
```

```
C
   20 A(1)=.5
      A(2)= 1.D0 - DSQRT( 0.5D0 )
      A(3)= 1.D0 + DSQRT( 0.5D0 )
      A(4)= 1.D0/6.D0
      B(1)=2.
      B(2)=1.
      B(3)=1.
      B(4)=2.
      C(1)=.5
      C(2)= A(2)
      C(3)= A(3)
      C(4)=.5
C
C*** PREPARATIONS OF FIRST RUNGE-KUTTA STEP
C
      DO 30 I=1,NDIM
      AUX(1,I)=Y(I)
      AUX(2,I)=DERY(I)
      AUX(3,I)=0.
   30 AUX(6,I)=0.
      IREC=0
      H=H+H
      IHLF=-1
      ISTEP=0
      IEND=0
C
C*** START OF A RUNGE-KUTTA STEP
C*** STEP = 2 * SPECIFIED STEP
C
   40 IF((X+H-XEND)*H)70,60,50
   50 H=XEND-X
   60 IEND=1
C
C*** RECORDING OF INITIAL VALUES OF THIS STEP
C
   70 CALL FCT(X,Y,DERY,PRMT)
      CALL OUTP(X,Y,DERY,IREC,NDIM,PRMT)
      IF(PRMT(5))400,80,400
   80 ITEST=0
   90 ISTEP=ISTEP+1
C
C*** START OF INNERMOST RUNGE-KUTTA LOOP
C
      J=1
  100 AJ=A(J)
      BJ=B(J)
      CJ=C(J)
      DO 110 I=1,NDIM
      R1=H*DERY(I)
      R2=AJ*(R1-BJ*AUX(6,I))
      Y(I)=Y(I)+R2
      R2=R2+R2+R2
  110 AUX(6,I)=AUX(6,I)+R2-CJ*R1
      IF(J-4)120,150,150
  120 J=J+1
      IF(J-3)130,140,130
  130 X=X+H/2.
  140 CALL FCT(X,Y,DERY,PRMT)
      GOTO 100
C
C*** END OF INNERMOST RUNGE-KUTTA LOOP
C
C*** TEST OF ACCURACY
C
  150 IF(ITEST)160,160,200
C
C*** IN CASE ITEST=0 THERE IS NO POSSIBILITY FOR TESTING OF ACCURACY
C*** IF(ITEST=0) RK STEP JUST PERFORMED WAS FOR TWICE THE SPECIFIED STEP
C
  160 DO 170 I=1,NDIM
  170 AUX(4,I)=Y(I)
      ITEST=1
      ISTEP=ISTEP+ISTEP-2
  180 IHLF=IHLF+1
      X=X-H
      H=H/2.
      DO 190 I=1,NDIM
      Y(I)=AUX(1,I)
      DERY(I)=AUX(2,I)
  190 AUX(6,I)=AUX(3,I)
      GOTO 90
C
C*** IN CASE ITEST=1 TESTING OF ACCURACY IS POSSIBLE ONLY IF EACH
C*** HALF OF THE INTERVAL IS DONE(I.E.,IFF ISTEP IS EVEN)
C
  200 IMOD=ISTEP/2
      IF(ISTEP-IMOD-IMOD)210,230,210
  210 CALL FCT(X,Y,DERY,PRMT)
      DO 220 I=1,NDIM
      AUX(5,I)=Y(I)
  220 AUX(7,I)=DERY(I)
      GOTO 90
C
C*** ORIGINAL VERSION; absolute error
C
C     COMPUTATION OF TEST VALUE DELT
C  230      DELT=0.                  !Good so far
C     DO 240 I=1,NDIM
C  240      DELT=DELT+AUX(8,I)*ABS(AUX(4,I)-Y(I))
C     IF(DELT-PRMT(4))280,280,250
C
```

```
C*** RELATIVE ERROR
C
  230 DELT = 0.
      DO 240 I=1,NDIM
      ARG = ABS(AUX(4,I) + Y(I))
      IF(ARG .EQ. 0.) arg = .25*ABS(AUX(4,I))
      IF(ARG .EQ. 0.) ARG = ERRSET !if here,aux(4,i)=y(i)=0.0; rer=0.
      RER = AUX(8,I)*ABS(AUX(4,I) - Y(I))/ARG
  240 DELT = dMAX1(DELT,RER)
      IF(DELT-PRMT(4)) 280,280,250
C
C*** ERROR IS TOO GREAT
C
  250 IF(IHLF-10)260,360,360
  260 DO 270 I=1,NDIM
  270 AUX(4,I)=AUX(5,I)
C     WRITE(5,1200) DELT
C 1200      FORMAT(' ?RKGST? -- ERROR TOO GREAT',G13.5)
      ISTEP=ISTEP+ISTEP-4
      X=X-H
      IEND=0
      GOTO 180
C
C*** RESULT VALUES ARE GOOD
C
  280 CALL FCT(X,Y,DERY,PRMT)
      DO 290 I=1,NDIM
      AUX(1,I)=Y(I)
      AUX(2,I)=DERY(I)
      AUX(3,I)=AUX(6,I)
      Y(I)=AUX(5,I)
  290 DERY(I)=AUX(7,I)
      CALL FCT(X-H,Y,DERY,PRMT)
      CALL OUTP(X-H,Y,DERY,IHLF,NDIM,PRMT)
      IF(PRMT(5))400,300,400
  300 DO 310 I=1,NDIM
      Y(I)=AUX(1,I)
  310 DERY(I)=AUX(2,I)
      IREC=IHLF
      IF(IEND)320,320,390
C
C*** INCREMENT GETS DOUBLED TO KEEP UP WITH ERROR HALVING
C
  320 IHLF=IHLF-1
      ISTEP=ISTEP/2
      H=H+H
C
C*** ALLOW THE PROGRAM TO EXPAND BEYOND ORIGINAL STEP SIZE SPECIFICATION
C*** UP TO THE MAXIMUM
C
      IF(IHLF+IHLFMX)40,330,330
  330 IMOD=ISTEP/2
      IF(ISTEP-IMOD-IMOD)40,340,40
C
C*** EXPAND H DUE TO LOW ERROR VALUE
C*** ONLY IF TWO CONSECUTIVE STEPS WITHOUT MENTION OF ERROR HAVE
C*** COMPLETED. FACTOR USED FOR EXPANSION (EXPAND) WAS SHOWN TO WORK
C*** REASONABLY WELL FOR A PERIODIC FUNCTION. VALUES AS HIGH AS
C*** EXPAND=.5 WILL PRODUCE GOOD RESULTS FOR MONOTONIC FUNCTIONS.
C
  340 IF(DELT-EXPAND*PRMT(4)) 350,350,40
  350 IHLF=IHLF-1
      ISTEP=ISTEP/2
      H=H+H
      GOTO 40
C
C*** RETURNS TO CALLING PROGRAM
C
  360 IHLF=11
      CALL FCT(X,Y,DERY,PRMT)
      GOTO 390
  370 IHLF=12
      GOTO 390
  380 IHLF=13
  390 CALL FCT(X,Y,DERY,PRMT)
      CALL OUTP(X,Y,DERY,IHLF,NDIM,PRMT)
  400 RETURN
      END
####
```

```
C
C     ..............................................................
C
C        SUBROUTINE RTMI
C
C...........................................................
c
c     This routine was originally supplied by Digital Equipment
c     Corporation as part of the Scientific Subroutine Package
c     available for RT-11 as part of the Fortran Enhancement
c     Package.  It was adopted for use in this package.
c
C...........................................................
c
C        PURPOSE
C           TO SOLVE GENERAL NONLINEAR EQUATIONS OF THE FORM FCT(X)=0
C           BY MEANS OF MUELLER-S ITERATION METHOD.
C
C        USAGE
C           CALL RTMI (X,F,FCT,XLI,XRI,EPS,IEND,IER)
C           PARAMETER FCT REQUIRES AN EXTERNAL STATEMENT.
C
C        DESCRIPTION OF PARAMETERS
C           X      - RESULTANT ROOT OF EQUATION FCT(X)=0.
C           F      - RESULTANT FUNCTION VALUE AT ROOT X.
C           FCT    - NAME OF THE EXTERNAL FUNCTION SUBPROGRAM USED.
C           XLI    - INPUT VALUE WHICH SPECIFIES THE INITIAL LEFT BOUND
C                    OF THE ROOT X.
C           XRI    - INPUT VALUE WHICH SPECIFIES THE INITIAL RIGHT BOUND
C                    OF THE ROOT X.
C           EPS    - INPUT VALUE WHICH SPECIFIES THE UPPER BOUND OF THE
C                    ERROR OF RESULT X.
C           IEND   - MAXIMUM NUMBER OF ITERATION STEPS SPECIFIED.
C           IER    - RESULTANT ERROR PARAMETER CODED AS FOLLOWS
C                     IER=0 - NO ERROR,
C                     IER=1 - NO CONVERGENCE AFTER IEND ITERATION STEPS
C                             FOLLOWED BY IEND SUCCESSIVE STEPS OF
C                             BISECTION,
C                     IER=2 - BASIC ASSUMPTION FCT(XLI)*FCT(XRI) LESS
C                             THAN OR EQUAL TO ZERO IS NOT SATISFIED.
C
C        REMARKS
C           THE PROCEDURE ASSUMES THAT FUNCTION VALUES AT INITIAL
C           BOUNDS XLI AND XRI HAVE NOT THE SAME SIGN. IF THIS BASIC
C           ASSUMPTION IS NOT SATISFIED BY INPUT VALUES XLI AND XRI, THE
C           PROCEDURE IS BYPASSED AND GIVES THE ERROR MESSAGE IER=2.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           THE EXTERNAL FUNCTION SUBPROGRAM FCT(X) MUST BE FURNISHED
C           BY THE USER.
C
```

```
C        METHOD
C           SOLUTION OF EQUATION FCT(X)=0 IS DONE BY MEANS OF MUELLER-S
C           ITERATION METHOD OF SUCCESSIVE BISECTIONS AND INVERSE
C           PARABOLIC INTERPOLATION, WHICH STARTS AT THE INITIAL BOUNDS
C           XLI AND XRI. CONVERGENCE IS QUADRATIC IF THE DERIVATIVE OF
C           FCT(X) AT ROOT X IS NOT EQUAL TO ZERO. ONE ITERATION STEP
C           REQUIRES TWO EVALUATIONS OF FCT(X). FOR TEST ON SATISFACTORY
C           ACCURACY SEE FORMULAE (3,4) OF MATHEMATICAL DESCRIPTION.
C           FOR REFERENCE, SEE G. K. KRISTIANSEN, ZERO OF ARBITRARY
C           FUNCTION, BIT, VOL. 3 (1963), PP.205-206.
C
C     ..................................................................
C
      SUBROUTINE RTMI(X,F,FCT,XLI,XRI,EPS,IEND,IER)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
C
C     PREPARE ITERATION
      IER=0
      XL=XLI
      XR=XRI
      X=XL
      TOL=X
      F=FCT(TOL)
      IF(F)1,16,1
    1 FL=F
      X=XR
      TOL=X
      F=FCT(TOL)
      IF(F)2,16,2
    2 FR=F
      IF(dSIGN(1.D0,FL)+dSIGN(1.D0,FR))25,3,25
C
C     BASIC ASSUMPTION FL*FR LESS THAN 0 IS SATISFIED.
C     GENERATE TOLERANCE FOR FUNCTION VALUES.
    3 I=0
      TOLF=100.*EPS
C
C
C     START ITERATION LOOP
    4 I=I+1
C
C     START BISECTION LOOP
      DO 13 K=1,IEND
      X=.5*(XL+XR)
      TOL=X
      F=FCT(TOL)
      IF(F)5,16,5
    5 IF(dSIGN(1.D0,F)+dSIGN(1.D0,FR))7,6,7
```

```
C
C     INTERCHANGE XL AND XR IN ORDER TO GET THE SAME SIGN IN F AND FR
    6 TOL=XL
      XL=XR
      XR=TOL
      TOL=FL
      FL=FR
      FR=TOL
    7 TOL=F-FL
      A=F*TOL
      A=A+A
      IF(A-FR*(FR-FL))8,9,9
    8 IF(I-IEND)17,17,9
    9 XR=X
      FR=F
C
C     TEST ON SATISFACTORY ACCURACY IN BISECTION LOOP
      TOL=EPS
      A=ABS(XR)
      IF(A-1.)11,11,10
   10 TOL=TOL*A
   11 IF(DABS(XR-XL)-TOL)12,12,13
   12 IF(DABS(FR-FL)-TOLF)14,14,13
   13 CONTINUE
C     END OF BISECTION LOOP
C
C     NO CONVERGENCE AFTER IEND ITERATION STEPS FOLLOWED BY IEND
C     SUCCESSIVE STEPS OF BISECTION OR STEADILY INCREASING FUNCTION
C     VALUES AT RIGHT BOUNDS. ERROR RETURN.
      IER=1
   14 IF(dABS(FR)-dABS(FL))16,16,15
   15 X=XL
      F=FL
   16 RETURN
C
C     COMPUTATION OF ITERATED X-VALUE BY INVERSE PARABOLIC INTERPOLATION
   17 A=FR-F
      DX=(X-XL)*FL*(1.+F*(A-TOL)/(A*(FR-FL)))/TOL
      XM=X
      FM=F
      X=XL-DX
      TOL=X
      F=FCT(TOL)
      IF(F)18,16,18
C
C     TEST ON SATISFACTORY ACCURACY IN ITERATION LOOP
   18 TOL=EPS
      A=ABS(X)
      IF(A-1.)20,20,19
   19 TOL=TOL*A
   20 IF(dABS(DX)-TOL)21,21,22
   21 IF(dABS(F)-TOLF)16,16,22
 C
 C     PREPARATION OF NEXT BISECTION LOOP
   22 IF(DSIGN(1.D0,F)+dSIGN(1.D0,FL))24,23,24
   23 XR=X
      FR=F
      GO TO 4
   24 XL=X
      FL=F
      XR=XM
      FR=FM
      GO TO 4
 C     END OF ITERATION LOOP
 C
 C
 C     ERROR RETURN IN CASE OF WRONG INPUT DATA
   25 IER=2
      RETURN
      END
####
```

```
      PROGRAM SDEGADIS2
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C     Program description:
C
C     SDEGADIS2 is a simplification of DEGADIS2 which performs the downwind
C     dispersion portion of the calculation for a steady state source
C     described by DEGADIS1.
C
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled 'Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
C
C     Disclaimer:
C
C     This computer code material was prepared by the University of
C     Arkansas as an account of work sponsored by the U. S. Coast Guard
C     and the Gas Research Institute.  Neither the University of Arkansas,
C     nor any person acting on its behalf:
C
C     a.  Makes any warranty or representation, express or implied,
C         with respect to the accuracy, completeness, or usefulness
C         of the information contained in this computer code material,
C         or that the use of any apparatus, method, numerical model,
C         or process disclosed in this computer code material may not
C         infringe privately owned rights; or
C
C     b.  Assumes any liability with respect to the use of, or for
C         damages resulting from the use of, any information,
C         apparatus, method, or process disclosed in this computer
C         code material.
```

```
C
C
C***************************************************************************
C***************************************************************************
C***************************************************************************
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
      include '($ssdef)'
C
      EXTERNAL PSS,PSSOUT,SSG,SSGOUT
C
      COMMON
     $/TITL/ TITLE
     $/GEN2/ DEN(5,igen)
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/ERROR/ SYOER,ERRP,SMXP,WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,
     $ WTRUH,WTDHG
     $/STP/ STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/com_ss/ ESS,SLEN,SWID,OUTCc,OUTSZ,OUTB,OUTL,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_fl/ cflag,clfl,cufl
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
C
C
C      DIMENSIONS/DECLARATIONS
C
      logical cflag

      real*4 tt1
      REAL*8 K,ML,L
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
      character*24 tinp,tsrc,tobs
      character*80 title(4)
      character*4 pound
      character*3 gas_name
C
      character*4 TR2,ER2,Sr3,SSD,TR3
C
```

```
      character*40 opnrup1
      character opnrup(40)
      EQUIVALENCE (OPNRUP(1),opnrup1)
C
      dimension prmt(22),y(6),dery(6),aux(8,6)
C
C.......................................................
C
C     DATA
C
      DATA POUND/'//  '/,POUNDN/-1.D-20/
C
      DATA TIME0/0./,NDIM/0/
C
      DATA TR2/'.TR2'/,ER2/'.ER2'/
      DATA Sr3/'.Sr3'/
      DATA SSD/'.SSD'/,TR3/'.TR3'/
C
C
C.......................................................
C
C     MAIN
C
      T1 = SECNDS(0.)
      istat = lib$date_time(TOBS)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
      read(5,1135) nchar,opnrup
 1135 format(q,40a1)
C
      opnrup1 = opnrup1(1:nchar) // ER2(1:4)
      CALL ESTRT2ss(OPNRUP1)
C
C*** GET THE COMMON VARIABLES CARRIED FROM DEGADIS1
C
      opnrup1 = opnrup1(1:nchar) // tr2(1:4)
      CALL STRT2(OPNRUP1,H_masrte,CCP)
C
      opnrup1 = opnrup1(1:nchar) // sr3(1:4)
      OPEN(UNIT=8,TYPE='NEW',NAME=OPNRUP1,
     $ CARRIAGECONTROL='FORTRAN')
c
      cflag = isofl.eq. 1.or. ihtfl.eq. 0
C
      WRITE(8,1119)
      if(cflag) then
            WRITE(8,1116) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1118)
      else
```

```
            WRITE(8,1115) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1117)
            endif
      WRITE(8,1119)
c
C
C
 1115 FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
     1      'Concentration',1x,'Density',2x,3x,'Gamma',4x,
     1      'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
     1      'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
     1      11x,11x,11x,11x,3x,'Width',3x,11x,9x,
     1      2(1pg9.3,'mole%',1x))
 1116 FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
     1      'Concentration',1x,'Density',2x,
     1      'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
     1      'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
     1      11x,11x,11x,3x,'Width',3x,11x,9x,
     1      2(1pg9.3,'mole%',1x))
 1117 FORMAT(1H ,4X,'(m)',4x,11x,
     1      2(1X,'(kg/m**3)',1x),12x,4x,'(K)',
     1      5(8X,'(m)'))
 1118 FORMAT(1H ,4X,'(m)',4x,11x,
     1      2(1X,'(kg/m**3)',1x),4x,'(K)',
     1      5(8X,'(m)'))
 1119 FORMAT(1H )
C
C
C
C...........................................................
C
C     STEADY STATE CALCULATIONS
C
C
      opnrup1 = opnrup1(1:nchar) // ssd(1:4)
c     OPEN(UNIT=9,TYPE='NEW',NAME=OPNRUP1)
C
C
      AGAIN = .FALSE.
C
C
      L = OUTL
      B = OUTB
      SZ0 = OUTSZ
      Erate = ESS
      QSTR0 = Erate/2.D0/L/B
      Cc = OUTCc
      wc = swcl
      wa = swal
      enth = senl
      rho = srhl
```

```
c
      if(cc.gt. ccp) then
            write(lunlog,1126) cc,ccp
 1126       format(/,' ',10('****'),/,' cc: ',1pg13.5,' is greater',
      1      ' than ccp: ',1pg13.5,/,' ',10('****'),/)
            cc =ccp
            endif
C
      ratio1= u0*z0/alpha1/ z0**alpha1 * cc /b/qstr0/l
      ratio = ratio1* sz0**alpha1 * (B + sqrtpi/2.D0*sy0er)
      if(ratio.lt. 1.D0) then
            sy0er = (1.D0/(RATIO1*sz0**alpha1) - b)*2.D0/sqrtpi
      else
            sz0 = (1.D0/((B+ sqrtpi/2.D0*sy0er)*ratio1))**(1.D0/alpha1)
      endif

      humsrc = (1.D0-wc-wa*(1.D0+humid))/wc
      call setden(wc,wa,enth)

      if(cflag) then
            call adiabat(2,twc,twa,gas_lfl,ya,clfl,r,w,t,tt)
            call adiabat(2,twc,twa,gas_ufl,ya,cufl,r,w,t,tt)
            endif

c
      cclay = cc/dellay
      call adiabat(0,wclay,walay,yclay,yalay,cclay,rholay,w,enthlay,t)
c
C
C*** let everyone know
C
      WRITE(lunlog,1170) L,B
      WRITE(lunlog,1180) QSTR0,SZ0
      write(lunlog,1185) wclay,walay,rholay,cclay,t
      write(lunlog,1186) wc,wa,rho,cc
c
 1170 FORMAT(' LENGTH: ',1pG13.5,' BEFF: ',1pG13.5)
 1180 FORMAT(' TAKEUP FLUX: ',1pG13.5,' SZ0: ',1pG13.5)
 1185 format(' wclay: ',1pg12.5,' walay: ',1pg12.5,
      1     ' rholay: ',1pg12.5,' Cclay: ',1pg12.5,/,
      1     ' temlay: ',1pg13.5)
 1186 format(' wc: ',1pg12.5,'  wa: ',1pg13.5,/,
      1     ' rho: ',1pg12.5,' Cc: ',1pg12.5)
C
C*** PREPARE FOR STEADY STATE INTEGRATION.
C
      PRMT(1) = L/2.D0
      PRMT(2) = 6.023D13
      PRMT(3) = STPP
      PRMT(4) = ERRP
      PRMT(5) = SMXP
      PRMT(6) = Erate
      PRMT(7) = Cc  ! OUTPUT
      PRMT(8) = B   ! OUTPUT
C
C*** PRMT(9) & PRMT(10) ARE CONSTANTS FOR D(SY) & D(SZ)
C
      PRMT(9) = Ce*sqrt(G*Z0/ALPHA1*GAMMAF) *GAMMAF/U0
      PRMT(10)= Z0**ALPHA*K*USTAR*ALPHA1 * ALPHA1/U0
C     PRMT(11)= NREC
C     PRMT(12)=
C     PRMT(13)=
      prmt(18)= u0*z0/alpha1
      prmt(19)= rhoa*k*ustar*alpha1
      prmt(20)= rholay
      prmt(21)= sz0
      prmt(22)= sz0
C
      Y(1) = rholay*prmt(18)*(SZ0/z0)**alpha1   ! rholay*ueff*heff
      Y(2) = SY0ER*sy0er
      Y(3) = B + sqrtpi/2.D0*sy0er
      Y(4) = 0.              ! added heat
      Y(5) = 0.              ! mass above UFL
      Y(6) = 0.              ! mass above LFL
C
      DERY(1) = WTSZP
      DERY(2) = WTSYP
      DERY(3) = WTBEP
      dery(4) = wtdh
      dery(5) = 1.D0
      dery(6) = 1.D0
C
c     NDIM = 4
      ndim=6        ! to integrate the mass above LFL and UFL
C
      WRITE(lunlog,1130)
 1130 FORMAT('  Entering Integration Step -- B > 0. ')
C
C*** PERFORM INTEGRATION
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,PSS,PSSOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(9,IHLF)
C
      NREC = INT(PRMT(11))
      WRITE(lunlog,1100)NREC
 1100 FORMAT(3X,'NUMBER OF RECORDS IN PSS = 'I10)
C
      IF(AGAIN) THEN
            Y(3) = Y(5)     ! mass above UFL
            Y(4) = Y(6)     ! mass above LFL
            GO TO 120
            ENDIF
C
C*** GAUSIAN COMPLETION OF THE INTEGRATION
C
C*** PSSOUT FORCES THE ABOVE INTEGRATION TO FINISH WHEN B<0 FOR THE
C***  FIRST TIME. THE STEP BEFORE THIS OCCURS IS RECORDED ON UNIT 7.
C***  THE STEP WHEN B GOES NEGATIVE IS CURRENTLY IN Y.
C
C*** THE CALCULATION METHOD CHANGES THE CURRENT VALUE OF SY TO A VALUE
C***  CALCULATED AS IF BEFF=SY RETAINING THE LAST VALUE OF Cc IN THE
C***  MATERIAL BALANCE.
C
      heat = Y(4)
      rholay = prmt(20)
      Cc = PRMT(7)
      rhouh = Y(1)
      SZ = ( rhouh/rholay/prmt(18) )**(1.D0/alpha1) * z0
      SYT = Erate*ALPHA1 *(Z0/SZ)**ALPHA/U0/SZ/Cc/SQRTPI
C
      XT = PRMT(12)
      XV = (SYT/RT2/DELTA)**(1.D0/BETA) - XT
C
C*** SET UP INTEGRATION FOR THE GAUSSIAN DISPERSION PHASE.
C
      do i=1,22
      prmt(i) = 0.D0
      enddo
c
      PRMT(1) = XT
      PRMT(2) = 6.023D23
      PRMT(3) = STPG
      PRMT(4) = ERRG
      PRMT(5) = SMXG
      PRMT(6) = Erate
      PRMT(7) = Cc           !-- OUTPUT
      PRMT(8) = XV
C     PRMT(9) = 'BLANK'
C     PRMT(10)= 'BLANK'
C     PRMT(11)= 'BLANK'
C     PRMT(12)= DIST AT COMPLETION -- OUTPUT
C
      prmt(18) = u0*z0/alpha1
      prmt(19) = rhoa*k*ustar*alpha1
      prmt(20) = rholay
      prmt(21) = sz
      prmt(22) = sz
c
      Y(1) = rhouh
      Y(2) = heat
      Y(3) = Y(5)           ! mass above UFL
      Y(4) = Y(6)          ! mass above LFL
C
      DERY(1) = wtruh
      dery(2) = wtdhg
      dery(3) = 1.D0
      dery(4) = 1.D0
C
c NDIM = 2
      ndim=4        ! to integrate the mass above LFL and UFL
C
      WRITE(lunlog,1140)
 1140 FORMAT(' Entering Gaussian Stage of Integration ')
C
C*** PERFORM INTEGRATION
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SSG,SSGOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(10,IHLF)
C
      NREC = INT(PRMT(11))
C
  120 CONTINUE
c
c*** summarize the information about the mass above the LFL,UFL
c
      write(8,8000) 100.*gas_ufl,100.*gas_lfl,Y(4)-Y(3),Y(4)
 8000 format(//,' For the UFL of ',1pg13.5,' mole percent, and',
     $' the LFL of ',1pg13.5,' mole percent:',
     $//,' The mass of contaminant between the UFL and LFL is:'
     $,1pg13.5,' kg.',/,' The mass of contaminant above the LFL is: ',
     $1pg13.5,' kg.')
C
C
C.............................................................
C
c     CLOSE(UNIT=9)
      CLOSE(UNIT=8)
C
      opnrup1 = opnrup1(1:nchar) // tr3(1:4)
      CALL TRANS(OPNRUP1)
C
      tt1 = t1
      T1 = SECNDS(tT1)/60.
      WRITE(lunlog,4000) TOBS
      WRITE(lunlog,4010) T1
 4000 FORMAT(//,'SDEGADIS2 -->',//,3X,'BEGAN AT ',A40)
 4010 FORMAT(3X,'*** ELAPSED TIME *** ',1pG13.5,' min')
C
      STOP
      END
####

 8 -- SYS$DEGADIS:SDEGADIS2.FOR                    20-OCT-1987 00:55:21
```

```
      function series(arg)
c
c*** This function estimates the infinite series used in the evaluation
c***  of the mass above a certain concentration level.
c
c***********************************************************************
c
c*** WARNING: This routine will overflow for arguments greater than 13.8
c
c***********************************************************************
c
      implicit real*8  (a-h,o-z), integer*4 (i-n)
c
      parameter (kmax= 100, error=1.d-4)
c
      common/alp/ alpha,alpha1
c
c*** initialize some variables
c
      pp = 1.D0/alpha1
      pprod = pp     ! for p*(p+1)*...*(p+k)
c
c*** calculate the first term
c
      coef = 2.D0/3.D0
      series = coef / pp * arg**1.5D0
      old = series
c
c
      do 200 kk=1,kmax
      pow = DBLE(kk) + 1.5D0
      plus = DBLE(kk+1)

      pprod = (pp + DBLE(kk)) * pprod
      coef = 2.D0 * plus**2 / DBLE(2*kk + 3)/ plus * coef

      series = series + coef / pprod * arg**pow

      denom = dmax1( (series+old), error)
      if( abs(series-old)/denom .le. error) return

      old = series
  200 continue
c
      write(6,900) arg
  900 format(' SERIES? Number of iterations too small for argument: ',
     $ 1pg13.5)
      call exit
      end
####
```

```
C.......................................................
C
C     TIME SORT SUPERVISOR
C
      SUBROUTINE SORTS(TABLE)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON
     $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $       Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $       Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $        TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $        TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/CNOBS/ NOBS
C
      DIMENSION TABLE(1)
C
      REAL*8 ML,K
C
      CALL GETTIM
C
C
C*** TABLE(I) VALUES
C***  I          PARAMETER
C***  --           ---------
C***  1  11  DIST         1 TO 10 CURRENT READ
c***  2  12  Yc
C***  3  13  Cc           11 TO 20 PREVIOUS READ
c***  4  14  rho
c***  5  15  gamma
c***  6  16  temp
C***  7  17  SZ
C***  8  18  SY
C***  9  19  B
C***  10 20  TS
C
C***  21     DISTO
C***  22     INTERPOLATION FRACTION
C
      DO 100 I = 1,NOBS
C
      IT = 0
C
      DO 105 J=1,20
```

```
 105 TABLE(J) = 0.
C
      II = NREC(I,1)

      if( ii .eq. 0) goto 130
c
c*** read first record
c
      read(9,*) (table(k1),k1=1,9)
      table(10) = ts( t0(I), table(1) )
      table(21) = table(1)
c
c*** loop through and read each record even if not pertinent
c
      DO 110 J = 2,II
C
             DO K1 = 1,10
             KK = K1 + 10
             TABLE(KK) = TABLE(K1)
             enddo
C
      READ(9,*) (TABLE(K1),K1=1,9)
      TABLE(10) = TS( T0(I), TABLE(1) )
C
      itl = int( (table(10)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 )
      itl = min( ntim, itl)
      itf = int( (table(20)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 ) + 1
      itf = max( 1, itf)

      do it = itf, itl, 1   ! do all points in range
C
C*** RECORD AN INTERPOLATED TIME SORTED POINT.
C
      KSUB(IT) = KSUB(IT) + 1
C
      TABLE(22) = (TIM(IT) - TABLE(20))/(TABLE(10) - TABLE(20))
C
      TDISTO(I,IT) = TABLE(21)
      TDIST(I,IT)  = TABLE(11) + (TABLE(1) - TABLE(11)) * TABLE(22)
      Tyc(I,IT)    = TABLE(12) + (TABLE(2) - TABLE(12)) * TABLE(22)
      TCc(I,IT)    = TABLE(13) + (TABLE(3) - TABLE(13)) * TABLE(22)
      Trho(I,IT)   = TABLE(14) + (TABLE(4) - TABLE(14)) * TABLE(22)
      Tgamma(I,IT) = TABLE(15) + (TABLE(5) - TABLE(15)) * TABLE(22)
      Ttemp(I,IT)  = TABLE(16) + (TABLE(6) - TABLE(16)) * TABLE(22)
      TSZ(I,IT)    = TABLE(17) + (TABLE(7) - TABLE(17)) * TABLE(22)
      TSY(I,IT)    = TABLE(18) + (TABLE(8) - TABLE(18)) * TABLE(22)
      TB(I,IT)     = TABLE(19) + (TABLE(9) - TABLE(19)) * TABLE(22)
C
      enddo
  110   CONTINUE
C
```

```
C
  130 II = NREC(I,2)
      IF(II .EQ. 0) GO TO 100

      DO 200 J=1,II
C
              DO K1 = 1,10
              KK = K1 + 10
              TABLE(KK) = TABLE(K1)
              enddo
C
      READ(9,*) (TABLE(K1),K1=1,7)
C
      TABLE(8) = RT2*DELTA*(TABLE(1) + XV(I))**BETA
      TABLE(9) = 0.
      TABLE(10) = TS(TO(I),TABLE(1))
C
      itl = int( (table(10)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 )
      itl = min( ntim, itl)
      itf = int( (table(20)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 ) + 1
      itf = max( 1, itf)

      do it = itf, itl, 1   ! do all points in range
C
C*** RECORD A TIME SORTED VALUE
C
      KSUB(IT) = KSUB(IT) + 1
      TABLE(22) = (TIM(IT) - TABLE(20))/(TABLE(10) - TABLE(20))
C
      TDISTO(I,IT) = TABLE(21)
      TDIST(I,IT)  = TABLE(11) + (TABLE(1) - TABLE(11)) * TABLE(22)
      Tyc(I,IT)    = TABLE(12) + (TABLE(2) - TABLE(12)) * TABLE(22)
      TCc(I,IT)    = TABLE(13) + (TABLE(3) - TABLE(13)) * TABLE(22)
      Trho(I,IT)   = TABLE(14) + (TABLE(4) - TABLE(14)) * TABLE(22)
      Tgamma(I,IT) = TABLE(15) + (TABLE(5) - TABLE(15)) * TABLE(22)
      Ttemp(I,IT)  = TABLE(16) + (TABLE(6) - TABLE(16)) * TABLE(22)
      TSZ(I,IT)    = TABLE(17) + (TABLE(7) - TABLE(17)) * TABLE(22)
      TSY(I,IT)    = TABLE(18) + (TABLE(8) - TABLE(18)) * TABLE(22)
      TB(I,IT)     = TABLE(19) + (TABLE(9) - TABLE(19)) * TABLE(22)
C
      enddo
  200 CONTINUE
C
  100 CONTINUE
C
      CALL SORTS1(TABLE)
C
      RETURN
      END
$$$$
```

```
C.............................................................
C
C
      SUBROUTINE SORTS1(TABLE)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON
     $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $      Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $      TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ALP/ ALPHA,alpha1
     $/CNOBS/NOBS
C
      DIMENSION TABLE(1)
C
      REAL*8 ML,K
      logical cflag
C
C*** DETERMINE IF ANY TIME VECTORS HAVE NO ENTRIES
C
      DO 192 I=1,NTIM
  192 IF(KSUB(I).GT. 2) GO TO 194 ! 1or2 points is of little value
      call trap(23)
  194 ISTRT = I
      DO 196 I=ISTRT,NTIM
  196 IF(KSUB(I).LE. 2) GO TO 198
      GO TO 199
  198 NTIM = I - 1
  199 CONTINUE
C
C*** REVERSE TIME SORTED VECTORS
C
C*** At this point, the results from Observer #1 is located in the
c***   first row of each vector.  But the information contributed by
c***   Observer #1 is the information which is the farthest (TDIST)
c***   from the source.  If the output is printed by time, the
```

```
c***   distances will decrease instead of increase.  So, we need to
c***   reverse each of the columns so that the downwind distances
c***   increase as you move down the column.  At the same time,
c***   ensure that all of the nonzero entries in a column are at
c***   the top of the column.
c
      DO 200 K1 = ISTRT,NTIM
C
      II = KSUB(K1)
      DO 170 J = 1,NOBS
C
c     IF(TDIST(J,K1).NE.0. .OR. TB(J,K1).NE.0.) GO TO 180
      IF( Tsz(J,K1).NE.0. ) GO TO 180
  170   CONTINUE
  180   II = II + J - 1
C
      DO 190 J = 1,II
C
      TABLE(II +        + 1 - J) =    TCc(J,K1)
      TABLE(II +   NOBS + 1 - J) =    Tyc(J,K1)
      TABLE(II + 2*NOBS + 1 - J) =   Trho(J,K1)
      TABLE(II + 3*NOBS + 1 - J) = Tgamma(J,K1)
      TABLE(II + 4*NOBS + 1 - J) =  Ttemp(J,K1)
      TABLE(II + 5*NOBS + 1 - J) =    TSY(J,K1)
      TABLE(II + 6*NOBS + 1 - J) =    TSZ(J,K1)
      TABLE(II + 7*NOBS + 1 - J) =     TB(J,K1)
      TABLE(II + 8*NOBS + 1 - J) = TDISTO(J,K1)
      TABLE(II + 9*NOBS + 1 - J) =  TDIST(J,K1)
C
  190   CONTINUE
C
      DO 210 J = 1,II
C
         TCc(J,K1) = TABLE(J)
         Tyc(J,K1) = TABLE(J +   NOBS)
        Trho(J,K1) = TABLE(J + 2*NOBS)
      Tgamma(J,K1) = TABLE(J + 3*NOBS)
       Ttemp(J,K1) = TABLE(J + 4*NOBS)
         TSY(J,K1) = TABLE(J + 5*NOBS)
         TSZ(J,K1) = TABLE(J + 6*NOBS)
          TB(J,K1) = TABLE(J + 7*NOBS)
      TDISTO(J,K1) = TABLE(J + 8*NOBS)
       TDIST(J,K1) = TABLE(J + 9*NOBS)
C
  210 CONTINUE
  200 CONTINUE
c
c
      if(sigx_flag.eq. 0.) then     ! no correction
c
            write(lunlog,*) ' No X-direction dispersion correction'
            DO 220 K1 = ISTRT,NTIM
            II = KSUB(K1)
            DO 220 I = 1,II
            TCcSTR(I,K1) = TCc(i,k1)
 220        CONTINUE
            goto 400
            endif
C
C*** GENERATE TCcSTR -- CENTER LINE CONCENTRATION CORRECTED FOR
C***  DOWNWIND DISPERSION.
C
      DO 230 K1 = ISTRT,NTIM
C
      II = KSUB(K1)
      DO 240 I = 1,II
C
c calculation for XP = TDIST(I,K1)
c
      TCcSTR(I,K1) = 0.
C
      DO 260 J = 1,II
C
      TABLE(J) = 0.
c     DIST = TDIST(J,K1) + RMAX
      DIST = TDIST(J,K1) - Tdist0(J,K1)
      deltax = ABS(tdist(i,k1) - tdist(j,k1))
c
      if(dist.lt. sigx_min_dist) then
            if(i.eq. j) then      ! i.e. deltax = 0.
             table(j) = (tdist(j+1,k1)- tdist(j-1,k1))/2.
             if(j.eq.1)table(j)= (tdist(2,k1)- tdist(1,k1))/2.
             if(j.eq. ii)table(j)=tdist(ii,k1)- tdist(ii-1,k1)
             table(j) = TCc(j,k1) / table(j) * RT2*SQRTPI
             endif
            goto 260
            endif
c
      SX = sigx_coeff* DIST**sigx_pow
C
      ARG = (deltax/SX)**2/2.
C
      IF(ARG .EQ. 0.) TABLE(J) = TCc(J,K1)/SX
      IF(ARG .NE. 0. .AND. ARG .LE. 80.)
     $                  TABLE(J) = TCc(J,K1)/SX/EXP(ARG)
C
 260  CONTINUE
C
      III = KSUB(K1)
      TCcSTR(I,K1) = TABLE(1)* (TDIST(2,K1)- TDIST(1,K1))/2.
      TCcSTR(I,K1) = TABLE(iii)* (TDIST(iii,K1)- TDIST(iii-1,K1))
      1                      +  TCcSTR(I,K1)
      iii = ksub(k1) - 1
C
      DO 280 J = 2,III
C
      TCcSTR(I,K1) = TABLE(J)* (TDIST(J+1,K1)- TDIST(J-1,K1))/2.
     1                        +  TCcSTR(I,K1)
C
  280   CONTINUE
c
      TCcSTR(I,K1) = TCcSTR(I,K1)/RT2/SQRTPI
c
c*** correct yc, rho, and temp values
c
            cc = Tccstr(i,k1)
      if(isofl.eq. 1 .or. ihtfl.eq. 0) then
            call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
      else
            enth = Tgamma(i,K1)
            call adiabat(-1,wc,wa,yc,ya,cc,rho,wm,enth,temp)
      endif
      Tyc(i,K1) = yc
      Trho(i,K1)= rho
C
  240   CONTINUE
C
  230   CONTINUE
C
C
C*** Estimate the mass between the UFL and LFL
C
 400  continue
c
      cflag = isofl.eq.1 .or. ihtfl.eq.0
      if(cflag) then
         call adiabat(2,aa,dd,gas_ufl,ee,chi ,gg,hh,pp,oo)
         call adiabat(2,aa,dd,gas_lfl,ee,clow,gg,hh,pp,oo)
      endif
c
      DO 430 K1 = ISTRT,NTIM
      kk  = k1+2*maxnob
      kk1 = k1+3*maxnob
      table(kk) = 0.
      table(kk1)= 0.
C
      II = KSUB(K1)
      DO 460 J = 1,II      ! evaluate the function at each point in space
C
c  initialize some values
c
      TABLE(J) = 0.
      jj = j+maxnob
      TABLE(JJ) = 0.
      cc = TCcstr(j,k1)
      bb = tb(j,k1)
      sy = tsy(j,k1)
      sz = tsz(j,k1)
      gamma = tgamma(j,k1)

      if(.not.cflag) then
         call adiabat(-2,aa,dd,gas_ufl,ee,chi ,gg,hh,gamma,oo)
         call adiabat(-2,aa,dd,gas_lfl,ee,clow,gg,hh,gamma,oo)
      endif
c
c*** Calculate the derivative for the total mass above the UFL and LFL
c
            gamhi  = 2.D0* Cc * Bb * Sz / alpha1
      gammax = 2.D0 *Cc *Sz *GAMMAF /alpha1 *(Bb +sqrtpi/2.D0 *Sy)

        if(cc.gt.clow) then
            wlow = Dlog(cc/clow)
            gamlow = gaminc(1.D0/alpha1, wlow ) * gamhi
            table(j)= gamlow + 2.D0*clow*Sy*Sz/alpha1*series(wlow)
            table(j)= DMIN1( table(j), gammax )
        endif

        if(cc.gt.chi) then
            whi  = Dlog(cc/chi )
            gamhi  = gaminc(1.D0/alpha1, whi  ) * gamhi
            table(jj) = gamhi+ 2.D0*chi *Sy*Sz/alpha1*series(whi)
            table(jj)= DMIN1( table(jj), gammax )
        endif
c
 460  continue
c
c*** now, finish the integration
c
      DO 450 J = 2,II        ! integrate in space at one value of time
      xx = (tdist(j,k1) - tdist(j-1,k1))
      arglow = (table(j) + table(j-1))/2.
      arghi  = (table(j+maxnob) + table(j-1+maxnob))/2.
      table(kk)  = arghi *xx + table(kk)
      table(kk1) = arglow*xx + table(kk1)
 450  continue
c
 430  continue
C
      RETURN
      END
$$$$
```

```
C     SOURCE EQUATIONS -- Gas Blanket present

      SUBROUTINE SRC1(time,Y,D,PRMT)


      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'

      parameter(    delt= 0.1D0,
     1      delto2= delt/2.D0,
     2      zero= 1.D-10,
     3      rcrit= 0.002D0)

      COMMON
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtUH,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/vucom/ vua,vub,vue,vud,vudelta,vuflag
     $/com_enthal/ h_masrte,h_airrte,h_watrte
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin

      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      logical vuflag

      REAL*8 ML,K
      REAL*8 L,masrte,mole
      INTEGER R,mass,massc,massa,ebal,mbal
      DIMENSION Y(7),D(7),PRMT(25)
      DATA R/1/,mass/2/,massc/3/,massa/4/,ebal/5/,mbal/6/


      if( prmt(20).lt. 0.D0) vuflag = .false.

      if(Y(mass) .le. 0.D0) then
            wc = dmax1(prmt(15),1.d-10)
            if(wc.gt. 1.) wc=1.d-10
            wa = 1.D0 - wc
            enthalpy = wc*h_masrte       ! air contributes nothing
      else
            wc = Y(massc)/Y(mass)
            wa = Y(massa)/Y(mass)
            enthalpy = Y(ebal)/Y(mass)
      endif

      humsrc = (1.D0 - wc - wa*(1.D0+humid))/wc
      call tprop(1,wc,wa,enthalpy,yc,ya,mole,temp,rho,cp)

      RADP = AFGEN2(PTIME,R1T,TIME,'R1TSRC')
      hei   = dmax1( Y(mass)/pi/Y(r)/Y(r)/rho , 0.0D0 )
      delrho = rho-rhoa
      gprime = g*delrho/rhoa *hei

C*** CALCULATE D(R),airrte,vel

      D(R) = 0.D0
      vel = 0.D0
      airrte = 0.D0
      Ri = 0.D0
      D(mbal)  = 0.D0

      IF(Gprime.GT. 0.D0) then
         slump = Ce*sqrt(Gprime)

         if(vuflag) then    ! momentum balance
            iii = 0        ! initialize loop counter
            vel = prmt(14)  ! old velocity value
            velmin = 0.D0
            velmax= dmax1( slump, 0.1D0, vel)

 100        hh = vel*vel/Ce/Ce/g/ (delrho/rhoa)
            rh = Y(r)-vua*vub*hh
            value = Y(r)**2/rh**2

               if(prmt(25).ge. prmt(24)) then   ! hh .ge. ht

            ht = 2.D0*(value*hei - vua*hh*(value-1.)) - hh
            velc = Y(mbal)/(0.4D0*pi*rho*(2.D0/3.D0*ht + hh)*rh**3/Y(r)
     1         + 2.D0/3.D0*pi*vua*rho*hh* (Y(r)**2 - rh*rh*rh/Y(r))
     2         + vue*pi*Y(r)*hei**2*rhoa)
            D(mbal) = pi*g*delrho*Y(r)*ht**2
     1         - vua*vud*pi*rhoa*Y(r)*hh*vel**2
               else

            ht = value*hei - vua*hh*(value-1.D0)
            velc = Y(mbal)/(2.D0/3.D0*pi*rho*ht*rh**3/Y(r)
     1         + 2.D0/3.D0*pi*vua*rho*hh* (Y(r)**2 - rh*rh*rh/Y(r))
     2         + vue*pi*Y(r)*hei**2*rhoa)
            D(mbal) = pi*g*delrho*(rh*ht**2 + vua*vub*hh*hh*hh)
     1         - vua*vud*pi*rhoa*Y(r)*hh*vel**2
               endif

            dif = abs(vel-velc)     ! convergence check
```

```
           sum = abs(vel) + abs(velc) + zero

               if(dif/sum .le. rcrit) then
            vel = (vel+velc)/2.D0
            prmt(13) = vel

                 if(vel .gt. 0.D0) then
                    Ri = gprime / vel**2
            airrte= 2.*pi* epsilon/Ri *rhoa*Y(r)*hei* vel
                    D(r) = vel
                    prmt(20) = slump
                 endif
                else

             dif = vel-velc

             if(velc.lt.velmin) velmin= dmax1(velc, 0.D0)
             if(velc.gt.velmax) velmax=velc

             if(dif .gt. 0.D0) then
                 velmax = vel
c                vel = 0.5D0*(velmax-velmin) + velmin
                 vel = 0.382D0*(velmax-velmin) + velmin
             else
                 velmin = velc
c                vel = (1.D0-0.5D0)*(velmax-velmin) + velmin
                vel = (1.D0-0.382D0)*(velmax-velmin) + velmin
             endif

             iii = iii+1
             if(iii .gt. 40) stop'SRC1 velocity loop'
             goto 100
                endif

          else

            vel= slump     ! gravity slumping
            hh = hei
            ht = hei
            Ri = gprime / vel**2
            airrte= 2.D0*pi* epsilon/Ri *rhoa*Y(r)*hei* vel
            D(r) = vel
          endif
      endif

c
      IF(delrho.Lt.delrhomin .and. .not.(check2 .or. u0.eq.0.))
     1     D(R) = 0. ! not for HSE types or no wind cases
c
      area = pi * (Y(r)**2 - radp**2)
      IF(Y(R).Le. RADP) THEN
```

```
            AREA = 0.D0
            Y(R) = RADP + zero
            IF(time.gt. delto2) then ! delt; num prob
             D(R) = dmax1(0.D0,
      1            ((AFGEN2(PTIME,R1T,TIME+delto2,'R1TSRD')-
      2      AFGEN2(PTIME,R1T, (TIME-delto2),'R1TSRe'))/ delt))
            else
             D(R) = dmax1(0.D0,
      1            ((AFGEN2(PTIME,R1T,delto2,'R1TSRD')-
      2             AFGEN2(PTIME,R1T, 0.D0,'R1TSRe'))/ delto2))
            endif
      ENDIF
c
c     calculate totrteout
c
      masrte = AFGEN2(PTIME,ET,TIME,'src1')
      PWCP = AFGEN2(PTIME,PWC,TIME,'src1')
      TOTPRT = MASRTE/PWCP
      HPRIM = AFGEN2(PTIME,PENTH,TIME,'src1')
      L = 2.0D0 * Y(R)
c
      cc = wc*rho
c
c
      qstrmx = 0.D0
      if(u0 .ne. 0.D0)
      1   qstrmx = cc*K*USTAR*ALPHA1*dellay/(dellay-1.D0)/phihat(rho,L)
c
c
c
      qstrll = qstrmx * pi*L*L/4.D0
      totrteout = qstrll/wc
c
c     surface effects
c
      watrte = 0.D0
      surface_q = 0.D0
      yw = 1.-ya-yc
      yw = min( max(yw, 0.0D0), 1.0D0)
      call surface(temp,hei,rho,mole,cp,yw,watrte,surface_q)
      surface_q = area * surface_q
      if(surface_q.lt. 0.D0) surface_q = 0.D0 ! don't let the cloud cool
      watrte = area * watrte
c
 500  totrtein  = airrte + TOTPRT + watrte
c
      IF(totrtein.lt.totrteout .and. .not.check2
      1      ) then ! check2 is True for HSE type spills
            D(R) = 0.
            if(hei.gt.srccut .and. Y(r).gt.srccut) then
            dHdt = (totrtein - totrteout)/3.D0/pi/ Y(r)/Y(r)/rho
```

```
             D(R) = Y(R)/Hei * dHdt
c                   ! Let cloud radius shrink when...
            endif           ! cloud height is decreasing unless...
      if( Y(R).le.0.01D0*rmax .and. masrte.eq.0.D0) D(R) = 0. ! the primary source...
            endif            ! has stopped. tsp6mar86
c
c     CALCULATE D(mass),D(massc),D(massa),D(anything left)
c
      D(mass)  = totrtein - totrteout
      D(massc) = masrte - qstrll
      D(massa) = (airrte + MASRTE*(1.D0/PWCP - 1.D0))/(1.D0+humid)
     $                      - wa/wc*qstrll
      D(ebal)  = 0.D0
      if(ihtfl.ne. 0)       ! equivalent to adiabatic mixing from TPROP for ihtfl=0
     $       D(ebal)  = HPRIM*TOTPRT + h_airrte*airrte
     $              + h_watrte*watrte - enthalpy*totrteout + surface_q
c
c
            uheff = qstrmx*L/cc
      sz = 0.D0
      if(u0 .ne. 0.D0) sz = ( uheff*alpha1/u0/z0 )**(1.D0/alpha1) * z0
c
c
C
      PRMT(6) = QSTRMX
      prmt(7) = sz
      prmt(8) = hei
      prmt(9) = rho
      prmt(10)= Ri
      prmt(11)= yc
      prmt(12)= ya
      prmt(13)= D(r)
      prmt(16) = wc
      prmt(17) = wa
      prmt(18) = enthalpy
      prmt(19) = temp
      prmt(21) = masrte
      prmt(22) = ht
      prmt(23) = hh
      RETURN
      END
c
c
C...........................................................
C
C     SUBROUTINE FOR OUTPUT FROM SOURCE in the presence of a Blanket
C
      SUBROUTINE SRC1O(TIME,Y,DERY,IHLF,NDIM,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
```

```
C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ ALPHA,alpha1
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      DIMENSION Y(6),DERY(6),PRMT(25)
      DIMENSION CURNT(iout_src),BKSP(iout_src),OUTP(iout_src)
C
      DATA I/0/,III/0/
      DATA EMAX/0./,tlast/0./
c
      REAL*8 ML,K
      INTEGER R,mass,massc,massa,ebal
      DATA R/1/,mass/2/,massc/3/,massa/4/,ebal/5/
c
      data nrec1/0/
C
      I = I + 1
      III = III + 1
C
      qstr = prmt(6)
      sz = prmt(7)
      hei = prmt(8)
      rho = prmt(9)
      Ri  = prmt(10)
      yc = prmt(11)
      ya = prmt(12)
      vel = prmt(13)
      prmt(14) = vel
      if(vel .gt. prmt(20)) prmt(20) = -prmt(20)
      prmt(15) = prmt(16)  ! wc
      wc = prmt(16)
      cc = wc * rho
      wa = prmt(17)
      enthalpy = prmt(18)
      temp = prmt(19)
      prmt(24) = prmt(22)  ! ht
      prmt(25) = prmt(23)  ! hh
c
      IF(hei .Le. 0.0D0) GO TO 1000
C
```

```
      QSAV = PI*Y(R)*Y(R)*qstr
      IF(QSAV .LT. EMAX) GO TO 110
      EMAX = QSAV
      RM = Y(R)
      SZM = SZ
  110   CONTINUE
      RMAX = dMAX1(RMAX,Y(R))
C
      IF(hei .Le. srccut) GO TO 1000
      if(cc .le. cclow .and. u0 .eq. 0.) goto 1000  ! no wind
      if(time.gt.tend+1. .and. u0.eq.0. .and. vel.eq.0.)goto 1000!no wind LNG
C
      IF(I .NE. 1) GO TO 115
      CURNT(1) = TIME
      CURNT(2) = Y(R)
      CURNT(3) = hei
      CURNT(4) = qstr
      CURNT(5) = sz
      CURNT(6) = yc
      CURNT(7) = ya
      CURNT(8) = rho
      CURNT(9) = ri
      CURNT(10)= wc
      CURNT(11)= wa
      CURNT(12)= enthalpy
      CURNT(13)= temp
      III = 1
      GO TO 125
  115 IF(I .EQ. 0) RETURN
C
      DO 116 II=1,iout_src
  116 BKSP(II) = CURNT(II)
C
      CURNT(1) = TIME
      CURNT(2) = Y(R)
      CURNT(3) = hei
      CURNT(4) = qstr
      CURNT(5) = sz
      CURNT(6) = yc
      CURNT(7) = ya
      CURNT(8) = rho
      CURNT(9) = ri
      CURNT(10)= wc
      CURNT(11)= wa
      CURNT(12)= enthalpy
      CURNT(13)= temp
C
      ERM = 0.
      ermss = 0.
      DO 120 II=2,iout_src
      div = curnt(ii)
      if(div .eq. 0.) div = srcoer
      ER1 = ABS( (CURNT(II)-BKSP(II))/div )
      ER2 = ABS( (CURNT(II)-OUTP(II))/div )
      if(II.ne.3 .and. ii.ne.9 .and. ii.ne.12 .and. ii.ne.7 .and. ii.ne.11)
      1     ermss = dMAX1(ER1,ER2,ERMss) ! ex hei,QSTR,Ri,enth,wa,ya for SS
  120 ERM = dMAX1(ER1,ER2,ERM)
C
      if(check4) then               ! steady state
            if( .not. (vel.eq. 0..and.time.gt.srcss)) goto 124
  122        check3 = .true.
             outcc = wc * rho
             swcl = wc
             swal = wa
             srhl = rho
             senl = enthalpy
             outl  = 2.0D0 * Y(r)
             Qstar = prmt(21)/pi/Y(r)**2
       if(u0.ne. 0.) sz= (alpha1/u0/z0*Qstar*outl/outcc)**(1.D0/alpha1)* z0
             outsz = sz
             outb  = pi*Y(r)**2 /outl/2.D0
             goto 1000

  124       if(ermss .gt. srcoer) goto 125

            if( time-tlast .gt. srcss) goto 122
       return
            endif
c
       IF(ERM .LT. SRCOER) RETURN
C
  125  CONTINUE
       tlast = time
       DO 130 II=1,iout_src
       IF(III.EQ.1) BKSP(II) = CURNT(II)
  130  OUTP(II) = BKSP(II)
C
       III = 0
       NREC1 = NREC1 + 1
       WRITE(9,2000) (OUTP(II),II=1,iout_src)
       RETURN
C
 1000    CONTINUE
       I = -1
       IF(TIME .GE. TEND) CHECK3 = .TRUE.
       NREC1 = NREC1 + 1
       WRITE(lunlog,1100)
       WRITE(lunlog,*) Hei,TIME
       TSC1 = TIME
       if(hei .le. 0.) then
             hei = 0.
             y(r) = dmin1(rmax,y(r))
            endif
        WRITE(9,2000)
        1     TIME,Y(R),hei,astr,sz,yc,ya,rho,ri,wc,wa,enthalpy,temp
        WRITE(lunlog,1110) NREC1
  C
        PRMT(5) = 1.
  C
        RETURN
   1100   FORMAT(5X,'VALUE OF Hei AT SOURCE TERMINATION -- @ TIME')
   1110   FORMAT(5X,'NUMBER OF LINES -->  ',I8)
   2000 format(1pg16.9,1x,1pg16.9,<iout_src-2>(1x,1pg13.6))
        END
####
```

```
      SUBROUTINE SRTOUT(OPNRUP, table)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $      Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SORTIN/TIM(maxnt),NTIM,ISTRT
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/alp/ alpha,alpha1
C
      dimension table(1)
c
      logical cflag,cflag1
c
      character*3 gas_name
      character*40 OPNRUP
C
      OPEN(UNIT=8,TYPE='NEW',NAME=OPNRUP,CARRIAGECONTROL='FORTRAN')
C
      WRITE(8,1100)
      if(sigx_flag.eq. 0.) then
            write(8,1102)
      else
            write(8,1104)
            write(8,1105) sigx_coeff,sigx_pow,sigx_min_dist
            endif
c
      cflag = isofl.eq. 1.or. ihtfl.eq. 0
      cflag1= isofl.eq.1
      if(cflag) then
            call adiabat(2,wc,wa,gas_lfl,ya,cc_lfl,r,w,t,tt)
            call adiabat(2,wc,wa,gas_ufl,ya,cc_ufl,r,w,t,tt)
            endif
C
      DO 110 I=ISTRT,NTIM
C
      WRITE(8,1119)
      WRITE(8,1119)
      WRITE(8,1110) TIM(I)
      if(cflag1) then
            WRITE(8,1116) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1118)
      else
            WRITE(8,1115) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1117)
            endif
      WRITE(8,1119)
      ip = 0
      II = KSUB(I)
c
      DO 120 J=1,II
c
      cc    = tccstr(j,i)
      rho   = Trho(j,i)
      yc    = Tyc(j,i)
      temp  = Ttemp(j,i)
      gamma = Tgamma(j,i)
      b     = tb(j,i)
      sz    = tsz(j,i)
      sy    = tsy(j,i)
      blfl  = 0.
      bufl  = 0.
c
      if(.not.cflag) then
            call adiabat(-2,wc,wa,gas_lfl,ya,cc_lfl,r,w,gamma,tt)
            call adiabat(-2,wc,wa,gas_ufl,ya,cc_ufl,r,w,gamma,tt)
            endif
c
      arg = (gas_zsp/sz)**alpha1
      if(arg .ge. 80.) goto 600
c
      ccz = cc/exp(arg)
      if(ccz .lt. cc_lfl) then
            if(cflag1) then
      WRITE(8,1120) TDIST(J,I),yc,Cc,rho,temp,B,SZ,SY
             else
      WRITE(8,1120) TDIST(J,I),yc,Cc,rho,gamma,temp,B,SZ,SY
             endif
            goto 600
            endif
      arg = -(dlog(cc_lfl/cc) + (gas_zsp/sz)**alpha1)
      blfl = sqrt(arg)*sy + b
c
      if(ccz .lt. cc_ufl) then
            if(cflag1) then
      WRITE(8,1120) TDIST(J,I),yc,Cc,rho,temp,B,SZ,SY,blfl
             else
      WRITE(8,1120) TDIST(J,I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
             endif
            goto 600
            endif
      arg = -(dlog(cc_ufl/cc) + (gas_zsp/sz)**alpha1)
      bufl = sqrt(arg)*sy + b
             if(cflag1) then
      WRITE(8,1120) TDIST(J,I),yc,Cc,rho,temp,B,SZ,SY,blfl,bufl
              else
      WRITE(8,1120) TDIST(J,I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl,bufl
              endif
c
  600 continue
      ip = ip + 1
      if(ip .eq. 3) then
            ip = 0
            write(8,1119)
            endif
  120 CONTINUE
c
c*** summarize the mass above the UFL and LFL
c
      aufl = table(i+2*maxnob)
      alfl = table(i+3*maxnob)
      write(8,8000) 100.*gas_ufl,100.*gas_lfl,alfl-aufl,alfl
 8000 format(//,' For the UFL of ',1pg13.5,' mole percent, and',
     $' the LFL of ',1pg13.5,' mole percent:',
     $//,' The mass of contaminant between the UFL and LFL is:'
     $,1pg13.5,' kg.',/,' The mass of contaminant above the LFL is: ',
     $1pg13.5,' kg.')
  110 CONTINUE
C
      CLOSE(UNIT=8)
C
C
 1100 FORMAT(1H0,5X,'Sorted values for each specified time.')
 1102 format(1H0,5x,'X-Direction correction was NOT applied.')
 1104 format(1H0,5x,'X-Direction correction was applied.')
 1105 format(1h ,5x,5x,'Coefficient:      ',1pg13.5,/,
      1     1h ,5x,5x,'Power:            ',1pg13.5,/,
      1     1h ,5x,5x,'Minimum Distance: ',1pg13.5' m')
 1110 FORMAT(1H0,5X,'Time after beginning of spill ',G14.7,' sec')
 1115 FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
      1     'Concentration',1x,'Density',2x,3x,'Gamma',3x,
      1     'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
      1     'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
      1     11x,11x,11x,11x,3x,'Width',3x,11x,9x,
      1     2(1pg9.3,'mole%',1x))
 1116 FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
      1     'Concentration',1x,'Density',2x,
      1     'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
      1     'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
      1     11x,11x,11x,3x,'Width',3x,11x,9x,
      1     2(1pg9.3,'mole%',1x))
 1117 FORMAT(1H ,4X,'(m)',4x,11x,
      1     2(1X,'(kg/m**3)',1x),11x,4x,'(K)',
     1     5(8X,'(m)'))
 1118 FORMAT(1H ,4X,'(m)',4x,11x,
     1     2(1X,'(kg/m**3)',1x),4x,'(K)',
     1     5(8X,'(m)'))
 1119 FORMAT(1H )
 1120 FORMAT(1H ,3(1X,1PG9.3,1X),2x,0pf7.4,2x,1X,1PG10.3,1X,
     1     6(1X,1PG9.3,1X))
C
      RETURN
      END
####
```

```
C..............................................................
C
C
      SUBROUTINE SSG(DIST,Y,Dery,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      DIMENSION Y(1),Dery(1),PRMT(1)
C
      include 'sys$degadis:DEGADIS2.dec'
c
      parameter (zero=1.D-10, rcrit=2.5D-3)
c
      COMMON
     $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
c
      REAL*8 K,ML

      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      INTEGER rhouh,dh, mhi, mlow
      DATA rhouh/1/,dh/2/, mhi/3/, mlow/4/
C
C*** PRMT(I) I/O
C***  I          VALUE          IN/OUT
C***   --        -----          ------
C***  6      E             IN
C***  7      Cc            OUT
C***  8      XV(I)         IN
C***  9      TO(I)         IN
C***  10     -             -
C***  11     NREC(I,2)     OUT -- STARTS OUTPUT UNIT=9
C***  12     DIST          OUT
c***  13     sz
c***  14     yc            out
c***  15     rho           out
c***  16     temp          out
c***  17     gamma         out
c***  18
c***  19
c***  20     rholay
c***  21     sz
C
```

```
      XV1 = PRMT(9)
      SY = RT2*DELTA*(DIST + XV1)**BETA
      Erate = PRMT(6)
c
c
      sz0 = prmt(22)
      sz = sz0
C
C*** MATERIAL BALANCE
C
      iii = 0
 100  Cc = Erate*ALPHA1*(Z0/SZ)**ALPHA/U0/SZ/SQRTPI/SY
c
      cclay = cc/dellay
      call addheat(cclay,Y(dh),rholay,temlay,cp)
      prod = dmax1( Y(rhouh)/rholay/prmt(18), zero)
      sz = ( prod ) **(1./alpha1) * z0
      dif = abs(sz - sz0)/(abs(sz)+abs(sz0)+zero)
      if(dif.gt. rcrit) then
            iii = iii+1
            if(iii.gt. 20) call trap(33)
            sz0 = sz
            goto 100
            endif
      prmt(20) = rholay
      prmt(21) = sz
      HEFF = GAMMAF*SZ/ALPHA1

      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
      call adiabat(0,wc,wa,yclay,ya,cclay,rholam,wml,enth,temlam)
      rit = 0.
c
      if(isofl.eq.0 .and. ihtfl.ne.0) then
            call addheat(cc,dellay*Y(dh),rho,temp,cp)
            rit = rift(temp,heff)
            endif
C
      RISTR = RIF(RHO,HEFF)
      PHI = PHIF(RISTR,rit)
C
      dery(rhouh) = prmt(19)/phi
      heigh = heff*dellay
      yw = 1.-yclay-ya
      yw = min( max( yw, 0.0D0), 1.0D0)
      call surface(temlay,heigh,rholay,wml,cp,yw,watrte,qrte)
      if(temp.ge. tsurf .or. temlay.ge. tamb) qrte = 0.
      dery(dh) = ( qrte/dellay-Y(dh)*Dery(rhouh) )/Y(rhouh)
C
c
c*** Calculate the derivative for the total mass above the UFL and LFL
c
```

```
      gamma = (rho-rhoa)/cc          ! gamma

      if(check4) then
        DERY(mlow) = 0.
        DERY(mhi)  = 0.
         if( isofl.eq.1 .or. ihtfl.eq.0 ) then
            call adiabat(2,aa,dd,gas_ufl,ee,chi ,gg,hh,pp,oo)
            call adiabat(2,aa,dd,gas_lfl,ee,clow,gg,hh,pp,oo)
         else
            call adiabat(-2,aa,dd,gas_ufl,ee,chi ,gg,hh,gamma,oo)
            call adiabat(-2,aa,dd,gas_lfl,ee,clow,gg,hh,gamma,oo)
         endif

      gammax = sqrtpi * Cc * Sz * Sy * GAMMAF / alpha1

        if(cc.gt.clow) then
            wlow = Dlog(cc/clow)
            DERY(mlow)= 2.D0*clow*Sy*Sz/alpha1*series(wlow)
            DERY(mlow)= DMIN1( DERY(mlow), gammax )
        endif

        if(cc.gt.chi ) then
            whi  = Dlog(cc/chi )
            DERY(mhi) = 2.D0*chi *Sy*Sz/alpha1*series(whi )
            DERY(mhi) = DMIN1( DERY(mhi ), gammax )
        endif
      endif
c
      PRMT(7) = Cc
      PRMT(12) = DIST
      prmt(14) = yc
      prmt(15) = rho
      prmt(16) = temp
      prmt(17) = gamma
      RETURN
      END
####
```

```
C.....................................................................
C
      SUBROUTINE SSGOUT(X,Y,D,IHLF,NDIM,PRMT)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      parameter (nsss=7, zero=1.e-10)
c
      DIMENSION Y(1),D(1),PRMT(1),BKSP(nsss),OUT(nsss),CURNT(nsss)
C
      COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/STOPIT/ TSTOP
c
      REAL*8 K,ML
C
C*** PARAMETER OUTPUT
C
C***  FROM SSG                  OUTPUT TO MODEL
C***  --------                  ---------------
C***  X              DIST
C***  PRMT(7)                 Cc
C***  Y(1)           SZ
c***  prmt(14)              yc
c***  prmt(15)              rho
C***  prmt(16)              temp
c***  prmt(17)              gamma
C
      ERM = 0.
      T01 = PRMT(9)
      TSL = TS(T01,X)
      prmt(22) = prmt(21)  ! sz
C
      IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR OUTPUT ROUTINE
C
      RII = -100./STPG
      RI = 0.
      CURNT(1) = X
      curnt(2) = prmt(14)  ! yc
      CURNT(3) = PRMT(7)   ! cc
      curnt(4) = prmt(15)  ! rho
      curnt(5) = prmt(17)  ! gamma
      curnt(6) = prmt(16)  ! temp
      CURNT(7) = prmt(21)  ! sz
C
   90 CONTINUE
C
C*** RECORD THE CURRENT AND PREVIOUS RECORDS
C
      RI = RI + 1.
C
      DO 100 II=1,nssg
 100  bksp(II) = curnt(II)
C
      CURNT(1) = X
      curnt(2) = prmt(14)  ! yc
      CURNT(3) = PRMT(7)   ! cc
      curnt(4) = prmt(15)  ! rho
      curnt(5) = prmt(17)  ! gamma
      curnt(6) = prmt(16)  ! temp
      CURNT(7) = prmt(21)  ! sz
C
c*** stop integration when cc<cclow and time is satisfied
C
      IF(PRMT(7).LT.CcLOW .AND. TSL.GE.TSTOP) GO TO 1000
C
C*** CHECK FOR OUTPUT
C
      DO 110 II=3,nssg
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
  110 ERM = dMAX1(ER1,ER2,ERM)
C
C*** OUTPUT RECORD IF ODLG IS EXCEEDED OR 100 METERS SINCE LAST OUTPUT
C
      DX = CURNT(1) - OUT(1)
      IF( RI.NE.1. .AND. ERM.LT.ODLG .AND. DX.LE.ODLLG) RETURN
C
C*** RECORD THE LAST POINT TO BE UNDER THE ERROR CRITERIA. IN CASE
C*** THE FIRST POINT AFTER A RECORD EXCEEDS THE ERROR BOUND, RECORD
C*** THAT POINT AS WELL.
C
      DO 120 II=1,nssg
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      WRITE(9,*) (OUT(II),II=1,nssg)
      RETURN
C
 1000 CONTINUE
C
```

```
C*** STOP INTEGRATION
C
      PRMT(12) = X
      TSTOP = TSL
      PRMT(11) = PRMT(11) + 1.
      WRITE(9,*) (CURNT(II),II=1,nsss)
C
      PRMT(5) = 1.
C
      RETURN
      END
####
```

```
C...............................................................
C
      SUBROUTINE SSGOUT(X,Y,D,IHLF,NDIM,PRMT)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      parameter (nssg=9, zero=1.e-10)
C
      DIMENSION Y(1),D(1),PRMT(1),BKSP(nssg),OUT(nssg),CURNT(nssg)
C
      include 'sys$degadis:DEGADIS2.dec/list'
C
      COMMON
     $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/ALP/ALPHA,alpha1
C
C
      REAL*8 ML,K
C
C*** PARAMETER OUTPUT
C
C***  FROM SSG                   OUTPUT TO MODEL
C***  --------                   ---------------
C***  X               DIST
C***  PRMT(7)                 Cc
C***  Y(1)            SZ
C***  PRMT(8)                 XV
C
      ERM = 0.
      prmt(22) = prmt(21)
C
      IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR OUTPUT ROUTINE
C
      RII = -100./STPG
      RI = 0.
      CURNT(1) = X
      CURNT(2) = PRMT(14)  ! yc
      CURNT(3) = prmt(7)   ! cc
      CURNT(4) = PRMT(15)  ! rho
      curnt(5) = prmt(17)  ! gamma
      curnt(6) = prmt(16)  ! temp
      curnt(7) = 0.0        ! b
      curnt(8) = prmt(21)  ! sz
      curnt(9) = rt2*delta*(x+prmt(8))**beta    ! sy
C
   90 CONTINUE
C
```

```
C*** RECORD THE CURRENT AND PREVIOUS RECORDS
C
      RI = RI + 1.
C
      DO 100 II=1,nssg
  100 BKSP(II) = CURNT(II)
      CURNT(1) = X
      CURNT(2) = PRMT(14)  ! yc
      CURNT(3) = prmt(7)   ! cc
      CURNT(4) = PRMT(15)  ! rho
      curnt(5) = prmt(17)  ! gamma
      curnt(6) = prmt(16)  ! temp
      curnt(7) = 0.0        ! b
      curnt(8) = prmt(21)  ! sz
      curnt(9) = rt2*delta*(x+prmt(8))**beta    ! sy
C
C*** STOP INTEGRATION WHEN Cc < CcLOW
C
      IF(PRMT(7).LT.CcLOW) GO TO 1000
C
C*** CHECK FOR OUTPUT
C
      DO 110 II=2,nssg
      if(ii.eq.7) goto 110
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
  110 ERM = dMAX1(ER1,ER2,ERM)
C
C*** OUTPUT RECORD IF ODLG IS EXCEEDED OR 100 METERS SINCE LAST OUTPUT
C
      DX = CURNT(1) - OUT(1)
      IF( RI.NE.1. .AND. ERM.LT.ODLG .AND. DX.LE.ODLLG) RETURN
C
C*** RECORD THE LAST POINT TO BE UNDER THE ERROR CRITERIA. IN CASE
C*** THE FIRST POINT AFTER A RECORD EXCEEDS THE ERROR BOUND, RECORD
C*** THAT POINT AS WELL.
C
      DO 120 II=1,nssg
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      call ssout(out)
      RETURN
C
 1000 CONTINUE
C
C*** STOP INTEGRATION
C
```

```
      PRMT(12) = X
      PRMT(11) = PRMT(11) + 1.
C
      call ssout(curnt)
C
      PRMT(5) = 1.
C
      RETURN
      END
####
```

```
      subroutine ssout(out)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension out(9)
c
      common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_fl/ cflag,clfl,cufl
     $/alp/ alpha,alpha1
c
      character*3 gas_name
c
      data ip/0/
c
      logical cflag
c
c
      dist = out(1)
      yc = out(2)
      cc = out(3)
      rho = out(4)
      gamma = out(5)
      temp = out(6)
      b = out(7)
      sz = out(8)
      sy = out(9)
c
      if(.not.cflag) then
            call adiabat(-2,wc,wa,gas_lfl,ya,clfl,r,w,gamma,tt)
            call adiabat(-2,wc,wa,gas_ufl,ya,cufl,r,w,gamma,tt)
            endif
c
      arg = (gas_zsp/sz)**alpha1
      if(arg .ge. 80.) goto 600
c
      ccz = cc/exp(arg)
      if(ccz .lt. clfl) then
            if(cflag) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY
             else
      WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY
             endif
            goto 600
            endif
      arg = -(dlog(clfl/cc) + (gas_zsp/sz)**alpha1)
      blfl = sqrt(arg)*sy + b
c
      if(ccz .lt. cufl) then
```

```
             if(cflag) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY,blfl
              else
      WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
              endif
             goto 600
             endif
      arg = -(dlog(cufl/cc) + (gas_zsp/sz)**alpha1)
      bufl = sqrt(arg)*sy + b
             if(cflag) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY,blfl,bufl
              else
      WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY,blfl,bufl
              endif
c
  600 continue
      ip = ip + 1
      if(ip .eq. 3) then
             ip = 0
             write(8,1119)
             endif
C
C
 1119 FORMAT(1H )
 1120 FORMAT(1H ,11(1X,1PG9.3,1X))
 1125 FORMAT(1H ,3(1X,1PG9.3,1X),2x,0PF7.4,2x,1X,1PG10.3,1X,
     1     6(1X,1PG9.3,1X))
C
      return
      end
$$$$
```

```
C...............................................................
C
C     PSEUDO-STEADY STATE SUPERVISOR
C
      SUBROUTINE SSSUP(H_masrte)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec/nolist'
c
      COMMON
     $/GEN3/ rads(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/gen2/ den(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ERROR/SY0ER,ERR0,SZ0ER,WTAI0,WTQ00,WTSZ0,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/nend/ poundn, pound
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/STOPIT/ TSTOP
     $/CNOBS/ NOBS
c
      REAL*8 K,ML,L
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      logical pup,pdn
c
      character*4 pound
      character*3 gas_name
c
      EXTERNAL PSS,PSSOUT,SSG,SSGOUT,OB,OBOUT
C
      DIMENSION PRMT(22),Y(5),DERY(5),AUX(8,5)
C
      DATA RTOT/0./
      data wma/28.96/, wmw/18.0/
C
c
c*** Estimate the earliest and latest time an observer can be released
```

```
c***   over the source.
c
      R = AFGEN(RADG, 0.0D0, 'RADG')
      T01 = TOOB(R, 0.0D0)
c
c**  For low wind speed cases which form a blanket, earlier times
c**    than T01 may be possible.  Check each of the points in RADG.
c
      do i=2,maxl
      if ( rads(1,i).eq.poundn .and. rads(2,i).eq.poundn) goto 20
      TOF = TOob( rads(2,i), rads(1,i) )
      if ( TOF .lt. T01) T01 = TOF
      enddo
c
c**  Now, calculate the last possible time an observer can be released.
c
 20   continue
      XEND = AFGEN(RADG,TEND,'RADG')
      TOF = TOOB(-XEND,TEND)
C
c*** Now, divide the total possible time among the observers.
c
      DTOB = (TOF-T01)/FLOAT(NOBS+1)
      T01 = T01 + DTOB
C
C*** perform the calculation for each observer
c
      write(12,1162)
c
      DO 120 I = 1,NOBS
C
C*** RESET AGAIN
C
      AGAIN = .FALSE.
C
      TO(I) = DTOB*dble(I-1) + T01
      pup = .true.
      pdn = .true.
C
C*** IF(XEND .GT. XIT(TEND,TO(I))) -- IS
C***  TRUE WHEN THE SOURCE WILL TERMINATE BEFORE THE OBSERVER
C***  CAN REACH THE DOWNWIND EDGE.
C
      IF(XEND .GT. XIT(TEND,TO(I))) then
            pdn = .false.
            TDOWN = TEND
            endif
c
C*** IF(XIT(0.0D0,TO(I)) .gt. -R) -- IS
C***  TRUE WHEN THE SOURCE WILL begin after THE OBSERVER
C***  has passed THE DOWNWIND EDGE @ t=0.0
```

```
C
      R = AFGEN(RADG,0.0D0,'RADG')
      IF(t0(i).le.0. .and. XIT(0.0D0,TO(I)).gt.-R) then
            pup = .false.
            TUP = 0.0D0
            endif
c
      if(pup) TUP = TUPF(TO(I))
      if(pdn) TDOWN = TDNF(TO(I))
C
      XDOWN = XIT(TDOWN,TO(I))
      XUP = XIT(TUP,TO(I))
      WRITE(lunlog,1160) TUP,XUP,TDOWN,XDOWN
C
C*** SET UP INTEGRATION PARAMETERS FOR EACH OBSERVER.
C
      do ijk=1,22
      prmt(ijk) =0.D0
      enddo
      do ijk=1,5
      y(ijk) = 0.D0
      dery(ijk)= 0.D0
      do ijkl=1,8
      aux(ijkl,ijk) = 0.D0
      enddo
      enddo
c
      PRMT(1) = TUP
      PRMT(2) = TDOWN
      PRMT(3) = STP0
      PRMT(4) = ERR0
      PRMT(5) = dMAX1(1.D0,(TDOWN-TUP)/50.D0)
      PRMT(6) = TO(I)
      prmt(7) = xup
      PRMT(13)= XDOWN - XUP
C
      Y(1) = sz0er  ! tsg3mar86
      Y(2) = sz0er  ! Mrate
      y(3) = sz0er * AFGEN(srcwc,tup,'IC')  ! Crate
      y(4) = sz0er * AFGEN(srcwa,tup,'IC') + 1.D-6  ! BDArate
      y(5) = sz0er * AFGEN(srcenth,tup,'IC')    ! Hrate
C
      DERY(1) = WTAIO
      DERY(2) = WTQOO
      DERY(3) = WTSZO
      DERY(4) = 1.D0
      DERY(5) = 1.D0
C
      NDIM = 4
      if(isofl.eq. 0 .and. ihtfl.ne. 0) ndim=5
C
```

```
C*** PERFORM INTEGRATION.
C
      WRITE(lunlog,1120) I
 1120 FORMAT(/,'  Entering Observer Integration Step for Observer # ',
     $       I3)
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,OB,OBOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(8,IHLF)
c
      write(lunlog,1125)
 1125 format(' ',10x,'Observer Integration complete...')
c
c
c*** Establish initial conditions
c
      cclay = prmt(14)
      cc = cclay*dellay
      wclay = prmt(15)
      walay = prmt(16)
      enthlay=prmt(17)
      rholay = prmt(18)
C
      L = XDOWN - XUP
      B = Y(1)
      AREA = B*L
      QSTR0 = Y(3)/area
      Erate = 2.D0*qstr0*L*b
      cc = min(cc, rhoe)
      WCP = AFGEN2(PTIME,PWC,TDOWN,'SSS-WC')
      RHOP = AFGEN2(PTIME,PRHO,TDOWN,'SSS-RH')
      CCP = WCP*RHOP
      if(cc .gt. ccp) cc = ccp

      sz0 = (qstr0*L/cc * alpha1/u0/z0)**(1.D0/alpha1) * z0
C
      ratio1= u0*z0/ALPHA1/ Z0**ALPHA1 *Cc /B/qstr0/L
      ratio = ratio1* sz0**alpha1 * (B + sqrtpi/2.D0*sy0er)
      if(ratio.le. 1.D0) then
            sy0er = (1.D0/(RATIO1*sz0**alpha1) - b)*2.D0/sqrtpi
      else
            sz0 = (1.D0/((B+ sqrtpi/2.D0*sy0er)*ratio1))**(1.D0/alpha1)
      endif

c
c*** Establish the thermodynamic properties of mixtures of air and
c***  the gas mixture (WCLAY,WALAY,ENTHLAY) assuming adiabatic
c***  mixing.  This is accomplished with the call to SETDEN.
c***  Then, extrapolate the properties to the centerline,
c***  ground level concentration.
c
```

```
      humsrc = (1.D0-wclay-walay*(1.D0+humid))/wclay
      call setden(wclay,walay,enthlay)
      if(isofl.eq. 1) goto 200
c
c*** Scan through the array DEN for the last value, and establish a new
c***  final value based on the centerline, ground level concentration.
c
      do iii= 1,igen
      if(den(1,iii) .gt. 1.D0) then
            ii = iii+1
            rholay = den(3,iii-1)
            temlay = den(5,iii-1)
            if(ii .gt. igen .or. iii.le.3) call trap(2)
            cc = cclay*dellay     ! exact due to profile assumptions
      if(cc.gt. rhoe) then
         write(lunlog,1126) cc,rhoe
 1126    format(/,' ',10('** **'),/,' cc: ',1pg13.5,' is greater',
     $        ' than rhoe: ',1pg13.5,/,' ',10('* ** *'),/)
         cc =rhoe
      endif
            rho = cc*(rholay-rhoa)/cclay + rhoa ! assumes gamma=con
            wc = cc/rho
            w2 = den(2,iii-2)/den(3,iii-2)
            w1 = den(2,iii-1)/den(3,iii-1)
            wa = (1.D0-(1.D0+humsrc)*wc)/(1.D0+humid)
            wm = 1.D0/(wc/gas_mw + wa/wma + (1.D0-wa-wc)/wmw)
            yc = wm/gas_mw * wc
            Yc= dmax1(0.0D0, dmin1(1.0D0, yc) )
            den(1,iii) = Yc
            den(2,iii) = cc
            den(3,iii) = rho
c
c*** now, determine the enthalpy and temperature of such a mixture.
c***  Base both enthalpy and temperature on the fact that
c***  (yc/rho) is proportional to temperature (and therefore enthalpy).
c
            denom = den(1,iii-2)/den(3,iii-2)-den(1,iii-1)/den(3,iii-1)
      if(denom.ne. 0.D0) then
         slope = (yc/rho-den(1,iii-1)/den(3,iii-1))/denom
         den(4,iii) = slope*(den(4,iii-2)-den(4,iii-1))+den(4,iii-1)
         den(4,iii) = dmin1( dmax1(h_masrte,den(4,iii)), 0.D0)
         den(5,iii) = slope*(den(5,iii-2)-den(5,iii-1))+den(5,iii-1)
         den(5,iii) = dmin1( dmax1(gas_temp,den(5,iii)), tamb)
      else
         den(4,iii) = den(4,iii-1)
         den(5,iii) = den(5,iii-1)
      endif
            temp = den(5,iii)
            den(1,ii) = 2.D0      ! end-of-record
c     if(cc.gt. rhoe) call trap(31)
            goto 200
```

```
                endif
        enddo
C
 200    CONTINUE

        IF(Cc .GT. RHOE) then
                WRITE(lunlog,1127) QSTRO,SZO,Cc,rhoe
 1127           format(/,' ',10('****  '),/,' qstr0: ',1pg13.5,
     $           ' sz0: ',1pg13.5,/,
     $           /,' cc: ',1pg13.5,' is greater',
     $            ' than rhoe: ',1pg13.5,//,' ',10('****  '),/)
c               call trap(30)
                endif
C
C*** SHOW THE OPERATOR WHAT IS GOING ON
C
        WRITE(lunlog,1160) TUP,XUP,TDOWN,XDOWN
        WRITE(lunlog,1170) AREA,L,B
        WRITE(lunlog,1180) QSTRO,SZO,sy0er
        write(lunlog,1185) wclay,walay,rholay,cclay,temlay
        write(lunlog,1186) wc,rho,cc,temp

        write(12,1161) i,t0(i), tup,tdown, xdown,L, B,Y(3), wc,temp,Sz0

 1160 FORMAT(/,' TUP: ',1pG13.5,' XUP: ',1pG13.5,' TDOWN: ',
     $ 1pG13.5,' XDOWN: ',1pG13.5)
 1161 format(' ',I3,1x,f8.1,1x,6(1x,f6.1,1x),1x,f5.3,1x,1x,
     $ f5.1,1x,1x,f5.2)
 1162 format(/,' ','obs',4x,'T0',4x,2x,'Tup',3x,1x,'Tdown',2x,
     $          1x,'Xdown',2x,1x,'Length',1x,1x,'HWidth',1x,1x,'E rate',1x,
     $          'Mass fr',1x,'Temp',2x,2x,'Sz0',//)
 1170 FORMAT(' Half AREA: ',1pG13.5,' LENGTH: ',1pG13.5,' B: ',1pG13.5)
 1180 FORMAT(' TAKEUP FLUX: ',1pG12.5,' SZ0: ',1pG12.5,
     $ ' sy0: ',1pg12.5)
 1185 format(' wclay: ',1pg12.5,' walay: ',1pg12.5,
     $          ' rholay: ',1pg12.5,' Cclay: ',1pg12.5,/,
     $          ' temlay: ',1pg13.5)
 1186 format(' wc: ',1pg12.5,
     $          ' rho: ',1pg12.5,' Cc: ',1pg12.5,' temp: ',1pg12.5)
C
C*** PREPARE FOR PSEUDO-STEADY STATE INTEGRATION.
C
        do ijk=1,22
        frmt(ijk) =0.D0
        enddo
        do ijk=1,5
        y(ijk) = 0.D0
        dery(ijk)= 0.D0
        do ijkl=1,8
        aux(ijkl,ijk) = 0.D0
        enddo
```

```
      enddo
C
      PRMT(1) = XDOWN
      PRMT(2) = 6.023D23
      PRMT(3) = STPP
      PRMT(4) = ERRP
      PRMT(5) = SMXP
      PRMT(6) = Erate
      PRMT(7) = Cc  ! -- OUTPUT
      PRMT(8) = B   ! -- OUTPUT
C
C*** PRMT(9) & PRMT(10) ARE CONSTANTS FOR D(SY) & D(SZ)
C
      PRMT(9) = Ce*sqrt(G*Z0/ALPHA1*GAMMAF)*GAMMAF/U0
      PRMT(10)= Z0**ALPHA*K*USTAR*ALPHA1*ALPHA1/U0
      PRMT(11)= NREC(I,1)
C     PRMT(12)= DIST AT COMPLETION -- OUTPUT
      PRMT(13)= T0(I)
c     prmt(14)= yc  ! output
c     prmt(15)= rho ! output
c     prmt(16)= temp       ! output; not recorded if isofl=1
c     prmt(17)= gamma      ! output; not recorded if isofl=1 .or. ihtfl=0
      prmt(18)= u0*z0/alpha1
      prmt(19)= rhoa*k*ustar*alpha1
      prmt(20)= rholay
      prmt(21)= sz0
      prmt(22)= sz0
C
      Y(1) = rholay*prmt(18)*(SZ0/z0)**alpha1       ! rho*ueff*heff
      Y(2) = SYOER*SYOER
      Y(3) = B + sqrtpi/2.D0*sy0er
      y(4) = 0.            ! added heat
C
      DERY(1) = WTSZP
      DERY(2) = WTSYP
      DERY(3) = WTBEP
      dery(4) = WTDH
C
      NDIM = 4
C
      WRITE(lunlog,1130)
 1130 FORMAT('  Entering Integration Step -- B > 0. ')
C
C*** PERFORM INTEGRATION
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,PSS,PSSOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(9,IHLF)
C
      NREC(I,1) = INT(PRMT(11))
      WRITE(lunlog,1100) NREC(I,1),T0(I)
 1100 FORMAT(3X,'NUMBER OF RECORDS IN PSS = 'I10,' FOR T0='1pg13.5)
C
      IF(AGAIN) GO TO 119
C
C*** GAUSIAN COMPLETION OF THE INTEGRATION
C
C*** PSSOUT FORCES THE ABOVE INTEGRATION TO FINISH WHEN B<0 FOR THE
C***  FIRST TIME. THE STEP BEFORE THIS OCCURS IS RECORDED ON UNIT 7.
C***  THE STEP WHEN B GOES NEGATIVE IS CURRENTLY IN Y.
C
C*** THE CALCULATION METHOD CHANGES THE CURRENT VALUE OF SY TO A VALUE
C***  CALCULATED AS IF BEFF=sqrtpi*SY/2. RETAINING THE LAST VALUE OF Cc IN THE
C***  MATERIAL BALANCE.
C
      heat = y(4)
      rholay = prmt(20)
      Cc = PRMT(7)
      rhouh = Y(1)
      sz = ( rhouh/rholay/prmt(18) )**(1.D0/alpha1) * z0
      SYT = Erate*ALPHA1*(Z0/SZ)**ALPHA/U0/SZ/Cc/SQRTPI
C
      XT = PRMT(12)
      XV(I) = (SYT/RT2/DELTA)**(1.D0/BETA) - XT
C
C*** SET UP INTEGRATION FOR THE GAUSSIAN DISPERSION PHASE.
C
      do ijk=1,22
      prmt(ijk) =0.D0
      enddo
      do ijk=1,5
      y(ijk) = 0.D0
      dery(ijk)= 0.D0
      do ijkl=1,8
      aux(ijkl,ijk) = 0.D0
      enddo
      enddo
c
      PRMT(1) = XT
      PRMT(2) = 6.023D23
      PRMT(3) = STPG
      PRMT(4) = ERRG
      PRMT(5) = SMXG
      PRMT(6) = Erate
      PRMT(7) = Cc  ! -- OUTPUT
      PRMT(8) = XV(I)
      PRMT(9) = T0(I)
C     PRMT(10)= 'BLANK'
      PRMT(11)= NREC(I,2)
C     PRMT(12)= DIST AT COMPLETION -- OUTPUT
c     prmt(13)= 'blank'
c     prmt(14)= yc         ! output
```

```
c      prmt(15)= rho         ! output
c      prmt(16)= temp         ! output
c      prmt(17)= gamma        ! output
       prmt(18)= u0*z0/alpha1
       prmt(19)= rhoa*k*ustar*alpha1
       prmt(20)= rholay
       prmt(21)= sz
       prmt(22)= sz
C
       Y(1) = rhouh
       Y(2) = heat
C
       DERY(1) = WTRUH
       dery(2) = WTDHG
C
       NDIM = 2
C
       WRITE(lunlog,1140)
 1140  FORMAT(' Entering Gaussian Stage of Integration ')
C
C*** PERFORM INTEGRATION
C
       CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SSG,SSGOUT,AUX)
C
       IF(IHLF .GE. 10) CALL trap(10,IHLF)
C
       NREC(I,2) = INT(PRMT(11))
       RTOT = RTOT + FLOAT(NREC(I,1) + NREC(I,2))
       WRITE(lunlog,1110) RTOT,I
 1110  FORMAT(5X,'TOTAL NUMBER OF RECORDS = ',1pG13.4,' THROUGH',
     $' OBS # ',I3)
C
       IF(RTOT .GT. 120000.) CALL trap(11)
C
  119  CONTINUE
       write(lunlog, 1150) tstop, prmt(12)
 1150  format(/,'  Last time Observer was active: ',1pg13.5,' s at ',
     $       1pg13.5,' m')
  120  CONTINUE
C
       RETURN
       END
####
```

```
C.............................................................
C
      SUBROUTINE STRT2(OPNRUP,H_masrte)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
C
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
C
      character*80 TITLE(4)
C
      character*4 pound
      character*24 TINP,TSRC,TOBS,TSRT
      character*3 gas_name
C
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      character*40 OPNRUP
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
      DO 90 I = 1,4
   90 READ(9,1000) TITLE(I)
 1000 FORMAT(A80)
C
      READ(9,*) NP
      DO 100 I=1,NP
  100 READ(9,*) PTIME(I),ET(I),R1T(I), PWC(I), PTEMP(I),
     $       PFRACV(I), PENTH(I), PRHO(I)
      PTIME(NP + 1)  = POUNDN
C
      READ(9,*) NP
      DO 220 I=1,NP
  220 READ(9,*) DEN(1,I),DEN(2,I),den(3,I),den(4,i),den(5,i)
      den(1,np+1) = 2.
C
      READ(9,*) NP
      DO 300 I=1,NP
      READ(9,*) rads(1,I),rads(2,I),qstr(2,I),srcden(2,I),srcwc(2,i),
     1      srcwa(2,i),srcenth(2,i)
      qstr(1,I) = rads(1,I)
      srcden(1,I) = rads(1,I)
      srcwc(1,i) = rads(1,i)
      srcwa(1,i) = rads(1,i)
      srcenth(1,i) = rads(1,i)
  300 continue
      I = NP + 1
      rads(1,I)  = POUNDN
      rads(2,I)  = POUNDN
      qstr(1,I) = POUNDN
      qstr(2,I) = POUNDN
      srcden(1,I) = POUNDN
      srcden(2,I) = POUNDN
      srcwc(1,I) = POUNDN
      srcwc(2,I) = POUNDN
      srcwa(1,I) = POUNDN
      srcwa(2,I) = POUNDN
      srcenth(1,I) = POUNDN
      srcenth(2,I) = POUNDN
C
      READ(9,1010) TINP,TSRC
      READ(9,1010) tobs,TSRT
C
      READ(9,*) U0,Z0,ZR,ML,USTAR
      read(9,*) K,G,RHOE,RHOA,DELTA
      read(9,*) BETA,GAMMAF,CcLOW
C
      READ(9,*) RM,SZM,EMAX,RMAX,TSC1
      read(9,*) ALEPH,TEND
C
      READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      READ(9,*) ALPHA
      alpha1 = alpha + 1.
C
      read(9,1020) gas_name
      read(9,*) gas_mw,gas_temp,gas_rhoe
      read(9,*) gas_cpk,gas_cpp
      read(9,*) gas_ufl,gas_lfl,gas_zsp
c
      read(9,*) istab
      read(9,*) tamb,pamb,humid
      humsrc = 0.D0
      read(9,*) isofl,tsurf
      read(9,*) ihtfl,htco
      read(9,*) iwtfl,wtco
c
      read(9,*) sigx_coeff,sigx_pow,sigx_min_dist
c
      read(9,*) iphifl,dellay
c
      H_masrte = 0.
      if(isofl.eq. 0) read(9,*) H_masrte
C
      READ(9,*) HTCUT, ce, delrhomin
c
 1010 format(2(a24,1x))
 1020 format(a3)
C
      CLOSE(UNIT=9)
C
      RETURN
      END
####
```

```
C..............................................................
C
      SUBROUTINE STRT2(OPNRUP,H_masrte,CCP)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      INCLUDE 'sys$degadis:DEGADIS2.DEC/LIST'
C
C........................................................
C
C     BLOCK COMMON
C
      COMMON
     $/TITL/ TITLE
     $/GEN2/ DEN(5,IGEN)
     $/ITI/ T1,TINP,TSRC,TOBS
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_ss/ ESS,SLEN,SWID,OUTCc,OUTSZ,OUTB,OUTL,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
C
      character*80 TITLE(4)
      character*24 TSRC,TINP,TOBS
      character*40 OPNRUP
      character*3 gas_name
      character*4 pound
C
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
      DO 90 I = 1,4
   90 READ(9,1000) TITLE(I)
 1000 FORMAT(A80)
C
      read(9,*) np
      do 100 i=1,np
      read(9,*) dummy1,dummy2,dummy3,PWC,PTEMP,PFRACV,PENTH,PRHO
  100 IF(I .EQ. 1) CCP = PWC*PRHO
c
      READ(9,*) NP
      DO 120 I=1,NP
  120 READ(9,*) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
      I = NP + 1
      DEN(1,I) = 2.
c
      read(9,*) np
      do 140 i=1,np
  140 read(9,*) dummy1,dummy2,dummy3,dummy4,dum5,dum6,dum7
C
      read(9,1100) tinp,tsc
      read(9,1100) tobs,tsrt
 1100 format(a24,1x,a24)
c
      READ(9,*) U0,Z0,ZR,ML,USTAR
      Read(9,*) K,G,RHOE,RHOA,DELTA
      read(9,*) BETA,GAMMAF,CcLOW
c
      read(9,*) dummy1
      read(9,*) dummy1
c
      READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
      READ(9,*) ALPHA
      alpha1 = alpha + 1.
c
      read(9,1200) gas_name
      read(9,*) gas_mw,gas_temp,gas_rhoe
      read(9,*) gas_cpk,gas_cpp
      read(9,*) gas_ufl,gas_lfl,gas_zsp
c
      read(9,*) istab
      read(9,*) tamb,pamb,humid
      humsrc = 0.D0
      read(9,*) isofl,tsurf
      read(9,*) ihtfl,htco
      read(9,*) iwtfl,wtco
      read(9,*) dummy1

      READ(9,*) ESS,SLEN,SWID
      read(9,*) OUTCc,OUTSZ,OUTB,OUTL
      read(9,*) swcl,swal,senl,srhl
C
      read(9,*) iphifl,dellay
      h_masrte = 0.
      if(isofl.eq. 0) read(9,*) H_masrte
C
      READ(9,*) HTCUT, ce, delrhomin
      CLOSE(UNIT=9)
      RETURN
 1200 format(a3)
      END
$$$$
```

```
C..........................................................
C
      SUBROUTINE STRT3(OPNRUP)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS3.dec/list'
C
C     BLOCK COMMON
C
      COMMON
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
      character*3 gas_name
      character*40 OPNRUP
      character*24 TINP,TSRC,TOBS,TSRT
C
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
      READ(9,*) NOBS
      DO 125 I=1,NOBS
  125 READ(9,*) NREC(I,1),NREC(I,2),TO(I),XV(I)
c
      READ(9,*) Npts
      DO 140 I=1,Npts
  140 READ(9,*) den(1,i),den(2,i),den(3,i),den(4,i),den(5,i)
      den(1,npts+1) = 2.
c
      READ(9,*) U0,Z0,ZR,ML,USTAR
      read(9,*) K,G,RHOE,RHOA,DELTA
      read(9,*) BETA,GAMMAF,CcLOW
c
      READ(9,1010) TINP,TSRC
      read(9,1010) TOBS,TSRT
```

```
 1010 format(2(a24,1x))
c
      READ(9,*) RM,SZM,EMAX,RMAX,TSC1
      read(9,*) ALEPH,TEND
c
      read(9,1020) gas_name
      read(9,*) gas_mw,gas_temp,gas_rhoe
      read(9,*) gas_cpk,gas_cpp
      read(9,*) gas_ufl,gas_lfl,gas_zsp
      read(9,*) istab
      read(9,*) tamb,pamb,humid
      humsrc = 0.D0
      read(9,*) isofl,tsurf
      read(9,*) ihtfl,htco
      read(9,*) iwtfl,wtco
      read(9,*) sigx_coeff,sigx_pow,sigx_min_dist
 1020 format(a3)
c
      READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      READ(9,*) ALPHA
      alpha1 = alpha + 1.
C
      CLOSE(UNIT=9)
C
      RETURN
      END
####
```

```
C     Surface effects
C
      SUBROUTINE Surface(temp,height,rho,mole,cp,yw,watrte,arte)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'
c
C
      COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/COM_SURF/ HTCUT
C
c
      REAL*8 ML,K
      REAL*8 L,masrte,mole
C
      vapor_p(txxx) = 6.0298e-3* exp(5407. *(1./273.15- 1./txxx))
c
c
c
      watrte = 0.
      arte = 0.
      if(isofl.eq.1 .or. ihtfl.eq.0) return
      if(height.le. htcut) return
      delta_t = tsurf - temp

      if(delta_t .lt. 0.) return
      top_vel = u0 * (height/z0)**alpha
      prod_nat = ((rho/mole)**2 * abs(delta_t)) ** 0.333333
      if(ihtfl .eq. 1) then       ! local correlation
            hn = 18. * prod_nat
            hf = 0.
            hf = 1.22 * rho*cp * (ustar/u0)**2 * top_vel
            ho = dmax1(hn,hf)
      else if(ihtfl .eq. 2) then    ! LLNL correlation
            ho = htco* rho* cp
      else if(ihtfl .eq. 3) then    ! Colenbrander's method
            av_temp= (tsurf + temp)/2.
            hn = 89.*(delta_t/av_temp**2)**.33333
            u10 = u0*(10./z0)**alpha
            hf = 1.22 * rho*cp * ustar**2/u10
            ho = dmax1(hn,hf)
      else
            ho = htco      ! ihtfl=-1
      endif
      arte = ho * delta_t
      if(arte .lt. 0.) arte = 0.
c                   ! since correlations are not valid for arte<0.
      watrte = 0.
      if(iwtfl .eq. 0) return
      fo = wtco
            if(iwtfl .gt. 0) then
            fn = 9.9e-3 * prod_nat
            ff = 20.7 * ho /cp /mole
            fo = dmax1(fn,ff)
            endif
      watrte = min( vapor_p(temp), yw*pamb )
      watrte = fo * (vapor_p(tsurf) - watrte)/pamb
      if(watrte .le. 0.) watrte = 0.
c
      return
      end
####
```

```
C
C     FUNCTION TO RETURN SZ0 CALCULATED over the source without
c      a blanket present underneath
c
c     NOTE: Uses the integration package RKGST and cannot be used
c         with any other routine without a local copy of RKGST.
C
      subroutine SZF(Q,L,WCP,sz,cclay,wclay,rholay)
c
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      external szlocal,szloco
C
      REAL*8 L
C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/szfc/ szstp0,szerr,szstpmx,szsz0
C
      dimension Y(1),D(1),PRMT(17),aux(8,1)
c
      prmt(1) = 0.
      prmt(2) = L
      prmt(3) = szstp0
      prmt(4) = szerr
      prmt(5) = szstpmx
      prmt(6) = Q
      PRMT(7) = WCP
c
      Y(1) = szsz0  ! rho*tdel*u0*z0/(1.+alpha)*(sz/z0)**(1.+alpha)
      D(1) = 1.
c
      ndim = 1
c
      call rkgst(prmt,y,d,ndim,ihlf,szlocal,szloco,aux)
c
      if(ihlf.ge. 10) call trap(3,ihlf)
c
      cclay = prmt(13)
      wclay = prmt(14)
      rholay = prmt(15)
      cc = prmt(16)
      sz = prmt(17)
C
      RETURN
      END
c
c
      subroutine szlocal(x,y,d,prmt)
c
```

```
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension y(1),d(1),prmt(1)
c
      common
     $/parm/ u0,z0,zr,ml,ustar,k,g,rhoe,rhoa,delta,beta,gammaf,cclow
     $/alp/ alpha,alpha1
     $/phicom/ iphifl,dellay
c
      real*8 ml,k
      integer rhouhlay/1/
c
      Q = prmt(6)
      WCP = PRMT(7)
      wclay = Q*x/Y(rhouhlay)
c
      call adiabat(1,wclay,walay,yc,ya,cclay,rholay,wm,enth,temp)
      cc = cclay*dellay
      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)  ! centerline
c
      uheff = Y(rhouhlay)/rholay/dellay
      sz = ( uheff/u0/z0*(alpha1) )**(1./alpha1) * z0
      heff = gammaf*sz/alpha1
      ristar = rif(rho,heff)
      phi = phif(ristar,0.D0)
      wel = dellay * k*ustar*alpha1/phi
      D(rhouhlay) = wel*rhoa + Q/WCP
c
      prmt(8) = cclay
      prmt(9) = wclay
      prmt(10)= rholay
      prmt(11)= cc
      prmt(12) = sz
      return
      end
c
c
      subroutine szloco(x,y, dery,ihlf,ndim, prmt)
c
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension y(1), dery(1), prmt(1)
c
      prmt(13) = prmt(8)
      prmt(14) = prmt(9)
      prmt(15) = prmt(10)
      prmt(16) = prmt(11)
      prmt(17) = prmt(12)
      return
      end
####
```

2 -- SYS$DEGADIS:SZF.FOR 20-OCT-1987 01:01:51

```
      subroutine tprop(ifl,wc,wa,enthalpy,yc,ya,wm,temp,rho,cp)
c
c     subroutine to return:
c            mole fractions (y's)
c            molecular weight (wm)
c            temperature (temp[=]K)
c            density (rho[=]kg/m**3)
c            heat capacity (cp[=]J/kg/K)
c
c     for a mixture from:
c            mass fractions (w's)
c            temperature (K)       for ifl.lt.0
c
c     for a mixture from:
c            mass fractions (w's)
c            enthalpy (J/kg)       for ifl.ge.0
c
c            adiabatic mixing of:  emitted gas @ gas_temp
c                        entrained ambient humid air @ tamb
c                        entrained water  from surface @ tsurf
c                  for ifl.eq.0 calculate and return
c
c            adiabatic lookup CALL ADIABAT
c                  for isofl.eq.1 .or. ihtfl.eq.0.and.ifl.eq.1
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'
c
      parameter (tfrac=0.618D0, tfrac1=1.D0-tfrac,
     .        rcrit=0.0005D0, acrit=1.D0, zero=1.D-20)
c
      common
     $/GEN2/ DEN(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
c
c*** data for air/water sys
c
      data wma/28.96D0/    ! molecular weight of air
      data wmw/18.02D0/    ! molecular weight of water
      data rho_water/1000.D0/     ! liquid water density [=] kg/m**3
      data cpa/1.0063D3/   ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/    ! heat capacity of water vapor[=]J/kg/K
      data dhvap/2.5023D6/ !latent heat of vap [=]J/kg water
      data dhfus/0.33D6/   !latent heat of fus [=]J/kg water
```

```
c
      logical rev
c
c
      vapor_p(txxx) = 6.0298D-3* exp(5407.D0 *(1.D0/273.15D0- 1.D0/txxx))
      sat_hum(p_tot,p_vp) = 0.622D0* p_vp/ (p_tot- p_vp) ! kg w/kg BDA
      ropw_air(p_tot,humxx) =
     ,          (.002833D0+ .004553D0*humxx)/p_tot/(1.D0+humxx) ! m**3/kg /K
c
c
      ww = 1.D0-wc-wa
      wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
      yc = wm/gas_mw *wc
      ya = wm/wma *wa
c
c
      if(isofl.eq. 1) then
            call adiabat(1,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
            return          ! interp density from wc
            endif
c
c
      if(ifl.eq. 0)
     $       enthalpy = wc*cpc(gas_temp)*(gas_temp - tamb)
     $       + (ww - wa*humid)*cpw*(tsurf - tamb)
     $       + wa*(1.D0+humid)*cpa*(tamb - tamb)    ! TR=tamb
c
c
c
      if(ifl.eq. 1 .and. ihtfl.eq.0) then
            call adiabat(1,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
            return          ! interp density from wc
            endif
c
c
      if(ifl .eq. -1) goto 400
c
c
      rev = .false.
 100  continue
      tmin = dmin1(gas_temp, tsurf, tamb)
      tmin0 = tmin
      tmax = dmax1(gas_temp, tsurf, tamb)
      tmax0 = tmax
      temp = (tmin+tmax)/2.D0
c
      do 300 j=1,35
      guess = enthal(wc,wa,temp)
      dif = enthalpy - guess
      sum = (abs(enthalpy) + abs(guess))/2.D0 + zero
      if(abs(dif)/sum.le.rcrit .or. abs(dif).le.acrit) goto 400
```

```
c
      if(dif.lt. 0.D0) then
            if(rev) tmax = temp
            if(.not.rev) tmin = temp
            temp = tmin + (tmax-tmin) * tfrac
      else
            if(rev) tmin = temp
            if(.not.rev) tmax = temp
            temp = tmin + (tmax-tmin) * tfrac1
      endif
c
 300  continue
      rev = .not.rev
      if(rev) goto 100
      write(lunlog,8050) wc,wa,enthalpy
 8050 format(' TPROP? wc: ',1pg12.5,1x,
      1     'wa: ',1pg12.5,1x,'enthalpy: ',1pg12.5)
      elow = enthal(wc,wa,tmin0)
      if(enthalpy.lt. elow) then  ! catch out of bounds numbers
            temp = tmin0
            enthalpy = elow
            goto 400
            endif
      elow = enthal(wc,wa,tmax0)
      if(enthalpy.gt. elow) then
            temp = tmax0
            enthalpy = elow
            goto 400
            endif
c
      call trap(24)
c
c
 400  continue               ! density calculation
      vp = vapor_p(temp)
      sat = dmax1( vp/pamb, 0.D0)
      conden = (ww - sat)/(1.D0 - sat)*wmw/wm
      conden = dmax1( 0.D0, conden)
      wwstar = (ww-conden)/(1.D0-conden)
      if(wa.gt.0.D0) sat = dmax1( wwstar/wa, 0.D0)
      rho = 1.D0/(temp*ropw_air(pamb,sat)*wa*(1.D0+sat)
     .        + wc*temp/gas_temp/gas_rhoe + conden/rho_water)
c
      tmin = temp + 10.
      if(tmin .gt. tmax0) tmin = temp - 10.
      if(tmin .lt. tmin0) tmin = temp + .1
c
      tmax = enthal(wc,wa,tmin)
      cp = (enthalpy - tmax)/(temp - tmin)
      if(cp .lt. cpa) cp = cpa      ! nominal value of air
c
```

```
      return
      end
c
c
      function cpc(temp)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
c
      data con/3.33D4/
c
      character*3 gas_name
c
      cpc = con
      if(temp .ne. gas_temp) then
            cpc = con + gas_cpk*
     .        (temp**gas_cpp - gas_temp**gas_cpp)/(temp- gas_temp)
            endif
      cpc = cpc/gas_mw
      return
      end
c
c
      function enthal(wc,wa,temp) ! used by TPROP
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      parameter (delta=10.D0)
c
      common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
c
      data cpa/1006.3D0/   ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/    ! heat capacity of water vapor[=]J/kg/K
      data dhvap/2.5023D6/ !latent heat of vap [=]J/kg water
      data dhfus/0.33D6/   !latent heat of fus [=]J/kg water
      data wma/28.96D0/    ! molecular weight of air
      data wmw/18.02D0/    ! molecular weight of water
c
c
      ww = 1.D0-wa-wc
      vp = 6.0298D-3* exp(5407.D0 *(1.D0/273.15D0- 1.D0/temp))
      sat = 0.622D0 * vp/ (pamb - vp) ! kg w/kg BDA
      wwstar = wa * sat
c
      dh = dhvap
      frac = 0.
      if(temp .lt. 273.15D0) frac = dmin1( (273.15D0-temp)/delta,1.D0)
      dh = dhvap + dhfus*frac
      wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
      sat = vp/pamb
      conden = (ww-sat)/(1.D0-sat)*wmw/wm
      conden = dmax1( 0.D0, conden)
c
 1000 enthal = wc*cpc(temp)*(temp - tamb)
     $       - conden*dh
     $        + (ww- wa*humid)*cpw*(temp - tamb)
     $        + wa*(1.D0+humid)*cpa*(temp - tamb)    ! TR=tamb
c
      return
      end
c
c
      subroutine adiabat(ifl,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
c
c     subroutine to return:
c           mass fractions (w's)
c           mole fractions (y's)
c           concentration (cc[=]kg/m**3)
c           density (rho[=]kg/m**3)
c           molecular weight (wm)
c           enthalpy ([=]J/kg)
c           temperature (temp[=]K)
c
c     for a mixture from DEN lookup of adiabatic mixing calculation
c     den(1,i)      mole fraction (yc)
c     den(2,i)      concentration (cc [=] kg c/m**3)
c     den(3,i)      mixture density (rho [=] kg mix/m**3)
c     den(4,i)      mixture enthalpy (enthalpy [=] J/kg)
c     den(5,i)      mixture temperature (temp [=] K)
c
c     ifl indicates given information:
c     -2)mole fraction (Yc) and assumption of constant gamma in enthalpy
c     -1)concentration (cc) and assumption of constant gamma in enthalpy
c     0) concentration (cc)
c     1) mass fraction c (wc)
c     2) mole fraction (Yc)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADISIN.dec'
```

```
c
      common
     $/GEN2/ DEN(5,igen)
     $/parm/ u0,z0,zr,ml,ustar,k,g,rhoe,rhoa,delta,beta,gammaf,cclow
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
      real*8 ml,k
c
c*** data for air/water sys
c
      data wma/28.96D0/     ! molecular weight of air
      data wmw/18.02D0/     ! molecular weight of water
c
c
      if(ifl.ne. 0) goto 1000
      ccl = cc
      if(cc .lt. 0.) ccl=0.
             i = 2
 30          if(den(1,i) .gt. 1.) then
              i=i-1
              if(cc.gt. den(2,i)) ccl=den(2,i)
              goto 50
              endif
             if(cc.le. den(2,i)) goto 50  ! lookup in concentration
             i=i+1
             goto 30
 50   slope = (den(3,i)-den(3,i-1)) / (den(2,i)-den(2,i-1)) ! interp in conc
             rho = (ccl - den(2,i-1))*slope + den(3,i-1)
      wcl = ccl / rho
      wc = wcl
      wa = (1.D0-(1.D0+humsrc)*wc)/(1.D0+humid)
      ww = 1.D0 - wa - wc
      wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
      yc = wm/gas_mw * wc
      ya = wm/wma * wa
      goto 8000
c
c
 1000 if(ifl.ne. -1) goto 1500
      ccl = cc
      if(ccl.lt. 0.) ccl=0.
      gamma = enthalpy
      wc = ccl/(rhoa+ccl*gamma)
      wa = (1.D0-(1.D0+humsrc)*wc)/(1.D0+humid)
      ww = 1.D0-wa-wc
      wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
      yc = wm/gas_mw * wc
      ya = wm/wma * wa
      rho = ccl/wc
      return
c
c
 1500 if(ifl.ne. -2) goto 1700
      ycl = yc
      if(ycl.lt. 0.) ycl=0.D0
      gamma = enthalpy
      ya = (1.D0-(1.D0+gas_mw*humsrc/wmw)*ycl)/(1.D0+humid*wma/wmw)
      yw = 1.D0-ya-ycl
      wm = ycl*gas_mw + ya*wma + yw*wmw
      wc = gas_mw/wm * ycl
      wa = wma/wm * ya
      cc = wc*rhoa/(1.D0 - gamma*wc)
      rho = cc/wc
      return
c
c
 1700 if(ifl.ne. 2) goto 2000
      ycl = yc
      if(yc .lt. 0.) then
          ycl = 0.
          wa = 1.D0/(1.D0+humid)
          ww = 1.D0-wa
          wm = 1.D0/(wma/wa + wmw/ww)
          ya = wm/wma * wa
          endif
      if(yc .gt. 1.) then
          ycl = 1.D0
          ya = 0.D0
          endif
      i = 2
 1730 if(den(1,i) .gt. 1.) then
            i = i-1
            goto 1750       ! extrapolate
            endif
      if(yc.le. den(1,i)) goto 1750 ! lookup in mole frac
      i=i+1
      goto 1730
 1750 slope = (den(2,i)-den(2,i-1)) / (den(1,i)-den(1,i-1)) ! interp in y
            cc = (ycl - den(1,i-1)) *slope + den(2,i-1)
      slope = (den(3,i)-den(3,i-1)) / (den(1,i)-den(1,i-1)) ! interp in y
            rho = (ycl - den(1,i-1))*slope + den(3,i-1)

      wc = cc/rho
      wm = ycl*gas_mw + ya*wma + (1.D0-ycl-ya)*wmw
      wa = ya*wma/wm

      i = 2
 1760 if(den(1,i) .gt. 1.) then
```

```
            i = i-1
            goto 1800       ! extrapolate
            endif
      cwc = den(2,i)/den(3,i)
      if(wc.le. cwc) goto 1800    ! lookup in mass frac
      i=i+1
      goto 1760
 1800       w1 = den(2,i-1)/den(3,i-1)
            w2 = den(2,i)/den(3,i)
      slope = (den(4,i)-den(4,i-1)) / (w2 - w1) ! interp in w
            enthalpy = (wc - w1) *slope + den(4,i-1)
      slope = (den(5,i)-den(5,i-1)) / (w2 - w1) ! interp in w
            temp = (wc - w1) *slope + den(5,i-1)
c
      return
c
c
 2000 if(ifl.ne. 1) goto 9000
      wcl = wc
          if(wc .lt. 0.) then
            wcl = 0.
            wa = 1.D0/(1.D0+humid)
            endif
          if(wc .gt. 1.) then
            wcl = 1.D0
            wa = 0.D0
            endif
      ww = 1.D0-wa-wcl
      wm = 1.D0/(wcl/gas_mw + wa/wma + ww/wmw)
      yc = wm/gas_mw *wcl
      ya = wm/wma *wa
      i = 2
 2030 if(den(1,i) .gt. 1.) then
            i = i-1
            goto 2050       ! extrapolate
            endif
      if(yc.le. den(1,i)) goto 2050 ! lookup in mole frac
      i=i+1
      goto 2030
 2050 slope = (den(3,i)-den(3,i-1)) / (den(1,i)-den(1,i-1))
      rho = (yc-den(1,i-1))*slope + den(3,i-1)
      slope = (den(2,i)-den(2,i-1)) / (den(1,i)-den(1,i-1))
      cc = (yc-den(1,i-1))*slope + den(2,i-1)
      i = 2
 2060 if(den(1,i) .gt. 1.) then
            i = i-1
            goto 8000       ! extrapolate
            endif
      cwc = den(2,i)/den(3,i)
      if(wcl.le. cwc) goto 8000   ! lookup in mass frac
      i=i+1
      soto 2060
c
c
 8000       w1 = den(2,i-1)/den(3,i-1)
            w2 = den(2,i)/den(3,i)
      slope = (den(4,i)-den(4,i-1)) / (w2 - w1) ! interp in w
            enthalpy = (wcl - w1) *slope + den(4,i-1)
      slope = (den(5,i)-den(5,i-1)) / (w2 - w1) ! interp in w
            temp = (wcl - w1) *slope + den(5,i-1)
c
      return
c
 9000 call trap(26)
      end
c
c
      subroutine setenthal(h_masrte,h_airrte,h_watrte)
c
c     subroutine to load /com_ENTHAL/ through passed arguments if needed
c
c


      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADISIN.dec'
c
      common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
c
c*** data for air/water sys
c
      data cpa/1.0063D3/   ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/    ! heat capacity of water vapor[=]J/kg/K
c
      h_masrte = 0.D0
      h_airrte = 0.D0
      h_watrte = 0.D0
c
      if(isofl.eq. 1) return
c
      h_masrte = cpc(gas_temp)*(gas_temp - tamb)    ! TR=tamb
c
c h_airrte = (1.+humid)*cpa*(tamb - tamb) = 0.
c
      if(iwatfl .eq. 0) return
      h_watrte = cpw*(tsurf - tamb)
```

```
c
      return
      end
c
c
c
      subroutine setden(wc,wa,enthalpy)
c
c     subroutine to load /GEN2/ as needed
c
c     adiabatic mixing of: WC
c                          WA
c                          WW @ specified enthalpy
c
c           with ambient humid air @ tamb
c
c     den(1,i)       mole fraction (yc)
c     den(2,i)       concentration (cc [=] kg c/m**3)
c     den(3,i)       mixture density (rho [=] kg mix/m**3)
c     den(4,i)       mixture enthalpy (enthalpy [=] J/kg)
c     den(5,i)       mixture temperature (temp [=] K)
c
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADISIN.dec'
c
      parameter (tcrit=0.002D0, zero=1.D-20)
      parameter (iils=200, ils=iils-1, iback=25)
c
      common
     $/GEN2/ DEN(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
c
      dimension curnt(5),backsp(5,iback)
c
c*** data for air/water sys
c
      data wma/28.96D0/     ! molecular weight of air
      data wmw/18.02D0/     ! molecular weight of water
      data cpa/1.0063D3/    ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/     ! heat capacity of water vapor[=]J/kg/K
c
c
c
```

```
      if(isofl.eq. 1) return
c
c
      k = 1
      den(1,k) = 0.0D0       ! yc
      den(2,k) = 0.0D0       ! cc
      den(3,k) = pamb*(1.D0+humid)/(.002833D0+ .004553D0*humid)/tamb ! rhoa
      den(4,k) = 0.0D0       ! enthalpy of ambient air; TR=tamb
      den(5,k) = tamb
c
c
      do 300 i= ils,1,-1
      zbda = (float(i)/float(iils)) / (1.+humid)
      zw = zbda*humid
      za = 1.D0-zbda-zw
c
c     enmix = za*enthalpy + zbda*(1.D0+humid)*cpa*tamb ! TR=tamb
      enmix = za*enthalpy
c
      zbda = zbda + za*wa
      za   =        za*wc
      call tprop(2,za,zbda,enmix,yc,ya,wm,temp,rho,cp)
      cc = za*rho
c
c
      curnt(1) = yc
      curnt(2) = cc
      curnt(3) = rho
      curnt(4) = enmix
      curnt(5) = temp
c
      if(i .eq. ils) then
            ind = 1
            do 150 jj= 1,5
 150        backsp(jj,ind) = curnt(jj)
            goto 300
            endif
c
c     ADIABAT interpolation scheme
c
      err = 0.
      do 180 iind = 1,ind
      yc    = backsp(1,iind)
      cc    = backsp(2,iind)
      rho   = backsp(3,iind)
      enmix = backsp(4,iind)
      temp  = backsp(5,iind)
      slope = (den(2,k)- curnt(2)) / (den(1,k)- curnt(1))
            ccint = (yc - curnt(1))*slope + curnt(2)
      err = dmax1(err,2.D0* abs(cc - ccint)/(abs(cc + ccint) + zero))
      slope = (den(3,k)- curnt(3)) / (den(1,k)- curnt(1))
```

```
          rhoint = (yc - curnt(1))*slope + curnt(3)
      err = dmax1(err,2.D0* abs(rho - rhoint)/(abs(rho + rhoint) + zero))
      wccal = cc / rhoint
          w1 = curnt(2)/curnt(3)
          w2 = den(2,k)/den(3,k)
      slope = (den(4,k)- curnt(4)) / (w2 - w1)
          entint = (wccal - w1)*slope + curnt(4)
      err = dmax1(err,2.D0* abs(enmix - entint)/(abs(enmix + entint) + zero))
      slope = (den(5,k) - curnt(5)) / (w2 - w1)
          temint = (wccal - w1)*slope + curnt(5)
      err = dmax1(err,2.D0* abs(temp - temint)/(abs(temp + temint) + zero))
 180  continue
c
      if(err .le. tcrit) then
          if(ind .ge. iback) goto 200
          ind = ind + 1
          do 190 jj=1,5
 190      backsp(jj,ind) = curnt(jj)
          goto 300
          endif
c
c     record a point in DEN
c
c
 200  k = k+1
      if(k.ge. igen) call trap(28)
      do 250 jj=1,5
      den(jj,k) = backsp(jj,ind)
 250  backsp(jj,1) = curnt(jj)
      ind = 1
c
 300  continue
c
      k = k+1
      if(k.ge. igen) call trap(28)
      if(wc.eq. 1.D00) then
          den(1,k) = 1.D00       ! yc
          den(2,k) = gas_rhoe    ! cc
          den(3,k) = gas_rhoe    ! rhoe
          den(4,k) = enthalpy    ! enthalpy
          den(5,k) = gas_temp    ! temp
      else
      call tprop(2,wc,wa,enthalpy,den(1,k),ya,wm,den(5,k),den(3,k),cp)
      den(2,k) = wc*den(3,k)       ! cc
      den(4,k) = enthalpy
      endif
      den(1,k+1) = 2.        ! .gt. 1. end-of-record indicator
c
      return
      end
c
```

```
c
c
c
      subroutine addheat(cc,dh,rho,temp,cp)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'
c
      parameter (tfrac=0.618D0, tfrac1=1.D0-tfrac,
     .        rcrit=0.005D0, acrit=1.D0, zero=1.D-20)
c
      common
     $/GEN2/ DEN(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
c
c*** data for air/water sys
c
      data wma/28.96D0/    ! molecular weight of air
      data wmw/18.02D0/    ! molecular weight of water
      data rho_water/1000.D0/     ! liquid water density [=] kg/m**3
      data cpa/1.0063D3/   ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/    ! heat capacity of water vapor[=]J/kg/K
      data dhvap/2.5023D6/ !latent heat of vap [=]J/kg water
      data dhfus/0.33D6/   !latent heat of fus [=]J/kg water
c
      logical rev
c
c
      vapor_p(txxx) = 6.0298D-3* exp(5407.D0 *(1.D0/273.15D0- 1.D0/txxx))
      sat_hum(p_tot,p_vp) = 0.622D0* p_vp/ (p_tot- p_vp) ! kg w/kg BDA
      ropw_air(p_tot,humxx) =
     .        (.002833D0+ .004553D0*humxx)/p_tot/(1.D0+humxx) ! m**3/kg /K
c
c
      cp = cpa
      rhoa = den(3,1)
c
      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enthalpy,amt)
      ww = 1.D0 - wc - wa
      temp = amt
      IF(isofl.eq.1 .or. ihtfl.eq.0) return ! adiabatic mixing is valid
      if(dh.le. 0.) return  ! catch colder surface temperatures
      enthalpy = enthalpy + dh
c
```

```
c
      if(enthalpy.gt. 0.) then
            temp = tamb
            goto 400
            endif
c
c
c
 100  continue
      tmin = amt    ! adiabatic mixing temp
      tmin0 = tmin
      tmax = dmax1(gas_temp, tsurf, tamb)
      tmax0 = tmax
      temp = (tmin+tmax)/2.D0
c
      do 300 j=1,35
      guess = enthal(wc,wa,temp)
      dif = enthalpy - guess
      sum = (abs(enthalpy) + abs(guess))/2.D0 + zero
      if(abs(dif)/sum.le.rcrit .or. abs(dif).le.acrit) goto 400
c
      if(dif.lt. 0.) then
            if(rev) tmax = temp
            if(.not.rev) tmin = temp
            temp = tmin + (tmax-tmin) * tfrac
      else
            if(rev) tmin = temp
            if(.not.rev) tmax = temp
            temp = tmin + (tmax-tmin) * tfrac1
      endif
c
 300  continue
      rev = .not.rev
      if(rev) goto 100
c     write(lunlog,8050) wc,wa,enthalpy,guess,temp
c8050 format(' ADDHEAT? wc: ',1pg12.5,1x,
c     1     'wa: ',1pg12.5,1x,'enthalpy: ',1pg12.5,/,
c     1     '        guess: ',1pg13.5,'  temp: ',1pg13.5)
      if(temp.lt. amt) call trap(17)
      elow = enthal(wc,wa,tmin0)
      if(enthalpy.lt. elow) then  ! catch out of bounds numbers
            temp = tmin0
            enthalpy = elow
            goto 400
            endif
      elow = enthal(wc,wa,tmax0)
      if(enthalpy.gt. elow) then
            temp = tmax0
            enthalpy = elow
            goto 400
            endif
```

```
c
      call trap(17)
c
c
  400 continue              ! density calculation
      vp = vapor_p(temp)
      sat = dmax1( vp/pamb, 0.D0)
      conden = (ww - sat)/(1.D0 - sat)*wmw/wm
      conden = dmax1( 0.D0, conden)
      wwstar = (ww-conden)/(1.D0-conden)
      if(wa .gt. 0.D0) sat = wwstar/wa
      rho = 1.D0/(temp*ropw_air(pamb,sat)*wa*(1.D0+sat)
     .        + wc*temp/gas_temp/gas_rhoe + conden/rho_water)
      if(temp.ne.amt) cp = dmax1(dh/(temp-amt),cpa)
c
c
c
      return
      end
$$$$
```

```
C.........................................................
C
C     FILE NAME TRANS1 -- FOR USE IN DEGADIS1
C
C.........................................................
C
      SUBROUTINE TRANS(FILE)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'
C
C     BLOCK COMMON
C
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/com_enthal/ h_masrte,H_airrte,H_watrte
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUTS
C
      character*80 TITLE(4)
C
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
      character*3 gas_name
C
      REAL*8 ML,K
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
```

```
      character*(*) file
C
      OPEN(UNIT=8,NAME=FILE,TYPE='NEW',
     $  carriagecontrol='list',
     $  recordtype='variable')
C
      WRITE(8,1000) (TITLE(I),I=1,4)
 1000 FORMAT(A80)
C
      DO 100 I=1,igen
  100 IF(PTIME(I).EQ.POUNDN) GO TO 105
      write(6,*) ' POUND WAS NOT DETECTED '
  105 NP = I - 1
      WRITE(8,1040) NP
      DO 110 I=1,NP
  110 WRITE(8,1030) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),
     $       PFRACV(I), PENTH(I), PRHO(I)
c
      DO 120 I=1,igen
  120 IF(DEN(1,I) .gt. 1.) GOTO 125
      DO 122 I=1,igen
  122 WRITE(8,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
      write(6,*) ' density function blew the loop'
  125 NP = I - 1
      WRITE(8,1040) NP
      DO 130 I=1,NP
  130 WRITE(8,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
c
      DO 140 I=1,maxl
c     cc = srcwc(2,i)*srcden(2,i)
c     if(cc.lt. cclow) then
c           fcc = 0.
c           do ii=i+1,maxl
c           fcc = amax1(srcwc(2,ii)*srcden(2,ii),fcc)
c           enddo
c           if(fcc.ge. cc) goto 140
c           np = i
c           tend = srcwc(1,i)
c           goto 146
c           endif
  140 IF(radg(1,I).EQ.POUNDN .AND. radg(2,I).EQ.POUNDN) GO TO 145
      write(6,*) ' POUND WAS NOT DETECTED '
  145 NP = I - 1
  146 WRITE(8,1040) NP
      DO 150 I=1,NP
  150 WRITE(8,1060) radg(1,i),radg(2,i),qstr(2,i),srcden(2,i),srcwc(2,i)
     1     ,srcwa(2,i),srcenth(2,i)
c
 1020 format(1x,i4,1x,1pg14.7)
 1030 format(8(1x,1pg14.7))
 1040 format(1x,i4)
 1050 format(2(a24,1x))
 1060 format(1x,1pg23.16,7(1x,1pg14.7))
 1070 format(1x,1pg14.7)
 1080 format(a3)
c
      WRITE(8,1050) TINP,TSRC
      write(8,1050) TOBS,TSRT
      WRITE(8,1060) UO,ZO,ZR,ML,USTAR
      write(8,1060) K,G,RHOE,RHOA,DELTA
      write(8,1030) BETA,GAMMAF,CcLOW
      WRITE(8,1060) RM,SZM,EMAX,RMAX,TSC1
      write(8,1030) ALEPH,TEND
      WRITE(8,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      WRITE(8,1070) ALPHA
      write(8,1080) gas_name
      write(8,1030) gas_mw,gas_temp,gas_rhoe
      write(8,1030) gas_cpk,gas_cpp
      write(8,1030) gas_ufl,gas_lfl,gas_zsp
      write(8,1040) istab
      write(8,1030) tamb,pamb,humid
      write(8,1020) isofl,tsurf
      write(8,1020) ihtfl,htco
      write(8,1020) iwtfl,wtco
      write(8,1030) sigx_coeff,sigx_pow,sigx_min_dist
C
      if(check4) then
            write(8,1030) ess,slen,swid
            write(8,1060) outcc,outsz,outb,outl
            write(8,1060) swcl,swal,senl,srhl
            end if
c
      write(8,1020) iphifl,dellay
c
      if(isofl.eq. 0) write(8,1030) H_masrte
c
      WRITE(8,1030) HTCUTS, ce, delrhomin
C
      CLOSE(UNIT=8)
C
      RETURN
      END
####
```

```
C............................................................
C
C     FILE NAME TRANS2 -- USE WITH DEGADIS2
C
      SUBROUTINE TRANS(FILE)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      COMMON
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/nend/ poundn,pound
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
c
      character*3 gas_name
      character*80 TITLE(4)
      character*24 TINP,TSRC,TOBS,TSRT
      character*(*) file
c
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      OPEN(UNIT=9,NAME=FILE,TYPE='NEW',
     $  carriagecontrol='list',
     $  recordtype='variable')
C
      WRITE(9,1040) NOBS
      DO 125 I=1,NOBS
  125 WRITE(9,1010) NREC(I,1),NREC(I,2),TO(I),XV(I)
c
      DO 140 I=1,igen
  140 IF(DEN(1,I).gt. 1.) GOTO 145
      write(6,*) ' density function error in TRANS'
  145 NP = I - 1
      WRITE(9,1040) NP
      DO 150 I=1,NP
  150 WRITE(9,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
C
```

```
      WRITE(9,1060) U0,Z0,ZR,ML,USTAR
      write(9,1060) K,G,RHOE,RHOA,DELTA
      write(9,1030) BETA,GAMMAF,CcLOW
c
      WRITE(9,1050) TINP,TSRC
      write(9,1050) TOBS,TSRT
c
      WRITE(9,1060) RM,SZM,EMAX,RMAX,TSC1
      write(9,1020) ALEPH,TEND
c
      write(9,1080) gas_name
      write(9,1030) gas_mw,gas_temp,gas_rhoe
      write(9,1020) gas_cpk,gas_cpp
      write(9,1030) gas_ufl,gas_lfl,gas_zsp
      write(9,1040) istab
      write(9,1030) tamb,pamb,humid
      write(9,1025) isofl,tsurf
      write(9,1025) ihtfl,htco
      write(9,1025) iwtfl,wtco
      write(9,1030) sigx_coeff,sigx_pow,sigx_min_dist
c
      WRITE(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
      WRITE(9,1070) ALPHA
c
 1010 format(1x,i8,1x,i8,2(1x,1pg14.7))
 1020 format(2(1x,1pg14.7))
 1025 format(1x,i4,1x,1pg14.7)
 1030 format(3(1x,1pg14.7))
 1040 format(1x,i4)
 1050 format(2(a24,1x))
 1060 format(5(1x,1pg14.7))
 1070 format(1x,1pg14.7)
 1080 format(a3,1x)
C
      CLOSE(UNIT=9)
      RETURN
      END
####
```

```
C...........................................................
C
C     FILE NAME TRANS2 -- USE WITH SDEGADIS2
C
C...........................................................
C
      SUBROUTINE TRANS(FILE)
C
C
C
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/ITI/ t1,TINP,TSRC,TOBS
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ALPHA,alpha1
C
      character*24 TSRC,TINP,TOBS
      character*3 gas_name
      character*(*) file
C
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      OPEN(UNIT=9,NAME=FILE,TYPE='NEW')
C
      WRITE(9,1060) U0,Z0,ZR,ML,USTAR
      write(9,1060) K,G,RHOE,RHOA,DELTA
      write(9,1030) BETA,GAMMAF,CcLOW
c
      WRITE(9,1050) TINP,TSRC
      write(9,1050) TOBS
c
      write(9,1080) gas_name
      write(9,1030) gas_mw,gas_temp,gas_rhoe
      write(9,1020) gas_cpk,gas_cpp
      write(9,1030) gas_ufl,gas_lfl,gas_zsp
c
      write(9,1040) istab
      write(9,1030) tamb,pamb,humid
      write(9,1025) isofl,tsurf
      write(9,1025) ihtfl,htco
      write(9,1025) iwtfl,wtco
c
```

```
      WRITE(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      WRITE(9,1070) ALPHA
C
      CLOSE(UNIT=9)
C
 1020 format(2(1x,1pg14.7))
 1025 format(1x,i4,1x,1pg14.7)
 1030 format(3(1x,1pg14.7))
 1040 format(1x,i4)
 1050 format(2(a24,1x))
 1060 format(5(1x,1pg14.7))
 1070 format(1x,1pg14.7)
 1080 format(a3,1x)
C
      RETURN
      END
####
```

```
C     FILE NAME TRANS3 FOR USE WITH DEGADIS3
C
C.............................................................
C
      SUBROUTINE TRANS(OPNRUP)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $      Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SORTIN/TIM(maxnt),NTIM,ISTRT
     $/ITI/T1,TINP,TSRC,TOBS,TSRT
C
      character*24 tinp,tsrc,tobs,tsrt
C
      character*(*) OPNRUP
C
      TO = TIM(ISTRT)
      DT = TIM(ISTRT+1) - TIM(ISTRT)
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='NEW',
     $      carriagecontrol='list',recordtype='variable')
C
C
      DO 110 I = ISTRT,NTIM
      II = I - ISTRT + 1
  110 KSUB(II) = KSUB(I)
      NTIM = NTIM - ISTRT + 1
      IF(NTIM .EQ. maxnt) GO TO 120
      II = NTIM + 1
      DO 115 I=II,maxnt
  115 KSUB(I) = 0
C
  120 CONTINUE
C
      WRITE(9,*) TO,DT,NTIM
      WRITE(9,*) KSUB
C
      CLOSE(UNIT=9)
C
      RETURN
      END
####
```

```
C.............................................................
C
C     SUBROUTINE TRAP -- DIAGNOSTICS
C
      SUBROUTINE trap(N,N1)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      COMMON /ITI/T1,TINP,TSRC,TOBS,TSRT

      real*4 tt1
c
      character*24 TINP,TSRC,TOBS,TSRT
c
      character*24 tt
      character*80 dd
C
      WRITE(lunlog,1100)
      WRITE(lunlog,1110)
      write(lunlog,1115) n
c
c*** check to see if the operator is ready to read the text of the error
c*** message
c
      write(lunlog,1000)
      irtn = lib$get_command( dd )  ! get a line from the terminal
 10   write(lunlog,1002)
C
      IF(N .EQ. 1 ) then
             WRITE(lunlog,2010) N1
             WRITE(lunlog,2011)
      else IF(N .EQ. 2 ) then
             WRITE(lunlog,2020)
      else IF(N .EQ. 3 ) then
             WRITE(lunlog,2030) N1
      else IF(N .EQ. 4 ) then
             WRITE(lunlog,2040)
      else IF(N .EQ. 5 ) then
             WRITE(lunlog,2050)
      else IF(N .EQ. 6 ) then
             WRITE(lunlog,2060)
      else IF(N .EQ. 7 ) then
             WRITE(lunlog,2070)
      else IF(N .EQ. 8 ) then
             WRITE(lunlog,2080) N1
             WRITE(lunlog,2081)
      else IF(N .EQ. 9 ) then
             WRITE(lunlog,2090) N1
```

```
            WRITE(lunlog,2091)
      else IF(N .EQ. 10) then
            WRITE(lunlog,2100) N1
            WRITE(lunlog,2101)
      endif
      IF(N .EQ. 11) WRITE(lunlog,2110)
      IF(N .EQ. 12) WRITE(lunlog,2120)
      IF(N .EQ. 13) WRITE(lunlog,2130)
      IF(N .EQ. 14) WRITE(lunlog,2140)
      IF(N .EQ. 15) WRITE(lunlog,2150)
      IF(N .EQ. 16) WRITE(lunlog,2160)
      IF(N .EQ. 17) WRITE(lunlog,2170)
      IF(N .EQ. 18) then
            WRITE(lunlog,2180) N1
            WRITE(lunlog,2181)
      endif
      IF(N .EQ. 19) WRITE(lunlog,2190) N1
      IF(N .EQ. 20) WRITE(lunlog,2200)
      IF(N .EQ. 21) WRITE(lunlog,2210)
      IF(N .EQ. 22) WRITE(lunlog,2220)
      IF(N .EQ. 23) then
            WRITE(lunlog,2230) maxnob
            WRITE(lunlog,2231)
      endif
      IF(N .EQ. 24) WRITE(lunlog,2240)
      IF(N .EQ. 25) WRITE(lunlog,2250)
      IF(N .EQ. 26) WRITE(lunlog,2260)
      IF(N .EQ. 27) WRITE(lunlog,2270)
      IF(N .EQ. 28) WRITE(lunlog,2280)
      IF(N .EQ. 29) WRITE(lunlog,2290)
      IF(N .EQ. 30) WRITE(lunlog,2300)
      IF(N .EQ. 31) WRITE(lunlog,2310)
      IF(N .EQ. 32) WRITE(lunlog,2320)
      IF(N .EQ. 33) WRITE(lunlog,2330)
C
 1000 format(/,5x,'Ready for the text of the error message? <CR> ',$)
 1002 format(/)
 1100 FORMAT(5X,'The best laid plans of mice and men...')
 1110 FORMAT(5X,'You have entered a TRAP -- the land of no RETURN.')
 1115 format(' Code: ',i4)

 2010 FORMAT(5X,'DEGADIS1? Source integration has returned IHLF=',I3,/,
     ./,'     This error occurs during integration of the equations',/,
     .' which describe the gas source.  IHLF is an error code',/,
     .' returned by the integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPIN in the ER1 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERBND in the ER1 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (one of the following: WTRG,',/,
     .' WTTM, WTYA, WTYC, WTEB, WTMB, or WTUH in the ER1 file).',/)
2011 format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPIN) is 0.  Correct the ER1 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPIN) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPIN must be positive.  Correct the ER1',/,
     .' file and execute the program again.',//)

2020 FORMAT(5X,'Reserved')

2030 format(5x,'SZF? Local integration failed; IHLF=',I3,/,
     ./,'     This error occurs during estimation of SZ over the',/,
     .' source when no gas is present.  IHLF is an error code',/,
     .' returned by the integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (SZSTPO in the ER1 file).  If this does not work,',/,
     .' increase the error criteria for all of the dependent',/,
     .' variables being integrated (SZERR in the ER1 file).',//,
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (SZSTPO) is 0.  Correct the ER1 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (SZSTPO) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  SZSTPO must be positive.  Correct the ER1',/,
     .' file and execute the program again.',//)

2040 format(5x,'SURFACE? Negative QRTE for positive DELTA_T',//,
     .'     This is a diagnostic message indicating an error in',/,
     .' estimation of the heat capacity.  Check the input',/,
     .' to the model and execute the program again.',//)

2050 FORMAT(5X,'CRFG? MORE POINTS FOR GEN3 WERE NEEDED',//,
     .'     The COMMON area /GEN3/ stores representative values',/,
     .' of the calculated source parameters.  If this message',/,
     .' occurs, relax the CRFG error criteria (CRFGER) in the',/,
     .' ER1 file.  If this is a common problem, the length of the',/,
     .' /GEN3/ vectors can be increased by changing the value of',/,
     .' MAXL in DEGADIS1.DEC and reinstalling DEGADIS.',//)

2060 FORMAT(5X,'TUPF? OBSERVER CALCULATIONS -- TUPF FAILED',//,
     .'      The trial and error search associated with finding the',/,
     .' upwind edge of the gas source for an observer failed.',/,
     .' Often, this problem can be avoided by adding one or two',/,
     .' additional observers to the present number of observers',/,
     .' (which changes the conditions for the trial and error).',/,
     .' Another possibility is to increase the error criteria for',/,
     .' this function (ERTUPF) in the ER2 file.',//)

 2070 FORMAT(5X,'TUPF? OBSERVER CALCULATIONS -- TDNF FAILED',//,
     .'      The trial and error search associated with finding the',/,
     .' downwind edge of the gas source for an observer failed.',/,
     .' Often, this problem can be avoided by adding one or two',/,
     .' additional observers to the present number of observers',/,
     .' (which changes the conditions for the trial and error).',/,
     .' Another possibility is to increase the error criteria for',/,
     .' this function (ERTDNF) in the ER2 file.',//)

 2080 FORMAT(5X,'SSSUP? OBSERVER INTEGRATION FAILED, IHLF=',I3,//,
     .'      This error occurs during integration of the five',/,
     .' differential equations which average the source for each',/,
     .' observer.  IHLF is an error code returned by the',/,
     .' integration package RKGST.',//,
     .'      When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPO in the ER2 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERRO in the ER2 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (one of the following: WTAIO,',/,
     .' WTQOO, or WTSZO in the ER2 file).',/)
 2081 format(
     .'      When IHLF=12, the initial increment of the independent',/,
     .' variable (STPO) is 0.  Correct the ER2 file and execute the',/,
     .' program again.',//,
     .'      When IHLF=13, the initial increment of the independent',/,
     .' variable (STPO) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPO must be positive.  Correct the ER2',/,
     .' file and execute the program again.',//)

 2090 FORMAT(5X,'SSSUP/SDEGADIS2? PSEUDO-STEADY INTEG FAILED, IHLF=',I3,
     .//,'      This error occurs during integration of the four',/,
     .' differential equations describing the portion of the',/,
     .' downwind calculation when b>0.  The routine calling TRAP is',/,
     .' SSSUP if a transient simulation is being executed; if a',/,
     .' steady state simulation is being executed, the calling',/,
     .' routine is SDEGADIS2.  IHLF is an error code returned by the',/,
     .' integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPP in the ER2 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERRP in the ER2 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (one of the following: WTSZP,',/,
     .' WTSYP, WTBEP, or WTDH in the ER2 file).',/)
 2091 format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPP) is 0.  Correct the ER2 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPP) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPP must be positive.  Correct the ER2',/,
     .' file and execute the program again.',//)

 2100 FORMAT(5X,'SSSUP/SDEGADIS2? GAUSSIAN INTEGRATION FAIL, IHLF=',I3,
     .//,'     This error occurs during integration of the',/,
     .' differential equations describing the portion of the',/,
     .' downwind calculation when b=0.  The routine calling TRAP is',/,
     .' SSSUP if a transient simulation is being executed; if a',/,
     .' steady state simulation is being executed, the calling',/,
     .' routine is SDEGADIS2.  IHLF is an error code returned by the',/,
     .' integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPG in the ER2 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERRG in the ER2 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (either WTRUH or WTDHG',/,
     .' in the ER2 file).',//)
 2101 format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPG) is 0.  Correct the ER2 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPG) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPG must be positive.  Correct the ER2',/,
     .' file and execute the program again.',//)

 2110 FORMAT(5X,'SSSUP/SDEGADIS2? TOTAL No. OF RECORDS EXCEED 120000',
     .//,'     This is an arbitrary stopping point for the process',/,
     .' in order to keep a runaway simulation from filling up disk',/,
     .' space.  Relax the output specifications (ODLP, ODLLP, ODLG,',/,
     .' or ODLLG) in the ER2 file in order to generate less output',/,
     .' if the input parameters are valid.',//)

 2120 FORMAT(5X,'Reserved')
 2130 FORMAT(5X,'Reserved')
 2140 FORMAT(5X,'Reserved')
 2150 FORMAT(5X,'Reserved')

 2160 FORMAT(5X,'PSSOUT/PSSOUTSS? PSS STARTED WITH B<0.',//,
     .'     This condition is checked at the beginning of the',/,
     .' downwind calculation in order to confirm proper handling of',/,
     .' the movement to the Gaussian phase of the downwind ',/,
     .' calculation.  Check the initial conditions and execute the',/,
     .' program again.',//)

 2170 format(5x,'TPROP/ADDHEAT? Enthalpy out of bounds',//,
     .'     Diagnostic message indication an enthalpy lower',/,
     .' than the adiabatic mixing enthalpy was passed to ADDHEAT.',/,
     .' Check the input conditions and execute the program again.',//)

 2180 FORMAT(5X,'ALPH? ALPHA INTEGRATION FAILED, IHLF=',I3,//,
     .'     The integration which determines the integral least',/,
     .' squares fit for ALPHA has failed.  Note that small values',/,
     .' of the Monin-Obukhov length ( ML < O(1m) ) in combination',/,
     .' with stable atmospheric conditions may cause this failure.',/,
     .' IHLF is an error code returned by the integration package',/,
     .' RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' absolute value of the initial step size of the independent',/,
     .' variable (STPINZ in the ER1 file).  If this does not work,',/,
     .' it will be necessary to increase the error criteria',/,
     .' (ERBNDZ in the ER1 file).',//)
 2181 format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPINZ) is 0. Correct the ER1 file and execute',/,
     .' the program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPINZ) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPINZ must be negative.  Correct the ER1',/,
     .' file and execute the program again.  This error will also',/,
     .' occur if the surface roughness ZR is greater than the',/,
     .' reference height Z0.',//)

 2190 FORMAT(5X,'ALPH? RTMI has failed to locate ALPHA IERR: ',I4,//,
     .'     The search procedure which determines ALPHA has failed.',/,
     .' This error may be the result of an unusual velocity',/,
     .' specification such as small values of the Monin-Obukvov',/,
     .' length ( ML < O(1.m) ) or small reference heights',/,
     .' ( ZO < O(10. * ML) ).  IERR is an error code returned by',/,
     .' the routine RTMI.',//,
     .'     When IERR=1, the search for ALPHA failed after a',/,
     .' specified number of iterations. Increase the error bound',/,
     .' used by RTMI (EPS in ER1 file).',//,
     .'     When IERR=2, the basic assumption that the function which',/,
     .' governs the search for ALPHA changes sign over the specified',/,
     .' interval is false.  Increase the search interval by',/,
     .' decreasing the lower bound of ALPHA (XLI in the ER1 file)',/,
     .' and increasing the upper bound (XRI in the ER1 file).',//)

2200 format(5x,'ESTRT? Premature EOF in RUN_NAME.ER1 or RUN_NAME.ER2.'
     .,//,'       The portion of the program which reads ER1 and',/,
     .' ER2 files encountered an end-of-file mark before all of',/,
     .' the information had been read.  Confirm these files and',/,
     .' execute the program again.  If necessary, copy and edit',/,
     .' the appropriate EXAMPLE file and execute the program again.',//)

2210 FORMAT(5X,'ESTRT1/ESTRT2/ESTRT2SS/ESTRT3? DECODE failed',//,
     .'     The portion of the program which reads the ER1, ER2,',/,
     .' or the ER3 file failed to understand a numerical entry.',/,
     .' The numbers must appear in columns 11-20 of the line with',/,
     .' no alphabetic characters in the field.  (Note that',/,
     .' exponential notation is not allowed.)  This restriction',/,
     .' does not apply to comment lines which have an exclamation',/,
     .' point (!) in the first column.',//)

2220 format(5x,'ESTRT1? The parameter file RUN_NAME.ER1 ',
     .'was not found.',//,
     .'     The ER1 file was not found for the current simualtion',/,
     .' (RUN_NAME).  Copy the file EXAMPLE.ER1 file to RUN_NAME.ER1',/,
     .' and edit it as necessary.  Execute the program again.',//)

2230 format(5x,'SORTS1? Fewer than 3 points sorted for any time.',//,
     .'     Only one or two simualtion points were applicable for',/,
     .' the sort times specified.  There are three possible causes',/,
     .' for this condition:',//,
     .'     (1) If this message appears when the sort times are',/,
     .' defaulted (CHECK5 is set to 0. in the ER3 file), the',/,
     .' number of observers will probably have to be increased',/,
     .' to give a good resolution of the downwind concentration',/,
     .' field.  (The number of observers is NOBS in the ER2 file',/,
     .' with a maximum of ',I3,' (MAXNOB) in DEGADIS2.DEC.)',/,
     .' As a rule of thumb, one gets good resolution of the downwind',/,
     .' concentration field if the ratio:',/,
     .' (secondary source duration / number of observers) is less',/,
     .' than about 10 seconds (or 20 at most).',//,
     .'     (2) The sort times specified in the ER3 file were',/,
     .' before the simulation had developed significantly.',/,
     .' This is only applicable when the user is specifying the sort',/,
     .' times (i.e. when CHECK5 is set to 1. in the ER3 file).',/,
     .' Increase the time of the first sort (ERT1), and rerun the',/,
     .' program.',/)
 2231 format(
     .'     (3) The sort times specified in the ER3 file were',/,
     .' after the gas was below the lowest concentration of interest',/,
     .' This is only applicable when the user is specifying the sort',/,
     .' times (i.e. when CHECK5 is set to 1. in the ER3 file).',/,
     .' Increase the time of the first sort (ERT1), and rerun the',/,
     .' program.  If additional results are desired for later',/,
     .' times, restart the simulation and specify a lower',/,
     .' concentration of interest in the input step ( lower',/,
     .' CCLOW in DEGADISIN).',//)

 2240 format(5x,'TPROP? Trial and error loop compromised',//,
     .'     TPROP estimates the temperature of a mixture based',/,
     .' upon the composition and enthalpy of the mixture.  Ensure',/,
     .' the properties for the diffusing species are entered',/,
     .' correctly and execute the simulation again.',//)

 2250 format(5x,'TPROP? Isothermal density loop compromised',//,
     .'     This error should never occur, but if it does, rebuild',/,
     .' the model from the original files and run the simulation',/,
     .' over.',//)

 2260 format(5x,'TPROP? Invalid entry flag in ADIABAT',//,
     .'     This is a programming diagnostic and should never occur.',/,
     .' If it does, rebuild the model from the original files.',//)

 2270 format(5x,'Reserved')

 2280 format(5x,'TPROP? IGEN request too large in SETDEN',//,
     .'     The subroutine SETDEN (in TPROP) performs a series of',/,
     .' adiabatic mixing calculations with a specified gas mixture',/,
     .' and ambient air and places the result in the array',/,
     .' DEN(5,IGEN).  This error indicates more points are needed in',/,
     .' DEN than were originally requested.  Increase the allocation',/,
     .' for DEN by changing the value of IGEN in DEGADISIN.DEC and',/,
     .' reinstalling DEGADIS.',//)

 2290 format(5x,'PHIF? flag IPHIFL is out of bounds',//,
     .'     Proper values of IPHIFL are integers between 1 and 5 ',/,
     .' inclusive.  Although values of IPHIFL are entered in the ER1',/,
     .' file as real numbers, they should be in this range.  Check',/,
     .' the ER1 file and execute the program again.',//)

 2300 format(5x,'SSSUP/SDEGADIS2? concentration greater than RHOE',//,
     .'     If the concentration of the contaminant becomes',/,
     .' greater than the pure component density for an isothermal',/,
     .' simulation, this error will occur.  However, this situation',/,
     .' should never occur.  Check the input conditions and execute',/,
     .' the program again.',//)

 2310 format(5x,'SSSUP? concentration greater than RHOE',//,
     .'     If the concentration of the contaminant becomes',/,
     .' greater than the pure component density for an isothermal',/,
     .' simulation, this error will occur.  However, this situation',/,
     .' should never occur.  Check the input conditions and execute',/,
     .' the program again.',//)

 2320 format(5x,'PSS? Sz convergence failure.',//,
     .'     This is a programming diagnostic and should never occur.',/,
     .' If it does, check the input conditions and execute the',/,
     .' program again.',//)

 2330 format(5x,'SSG? Sz convergence failure.',//,
     .'     This is a programming diagnostic and should never occur.',/,
     .' If it does, check the input conditions and execute the',/,
     .' program again.',//)
C
      CLOSE(UNIT=9)
C
      CALL TRANS('trap.DBG')
C
      istat = lib$date_time(TT)
      tt1 = t1
      ttime = secnds(tt1)/60.
C
  140 WRITE(lunlog,3000) TT
      WRITE(lunlog,3010) Ttime
 3000 FORMAT(1X,' -- ENDing AT ',A24)
 3010 FORMAT(5X,' ***** ELAPSED TIME ***** ',1pg13.5,' MIN ')
C
      irtn = LIB$DO_COMMAND( 'Exit' )    ! this issues the command EXIT to VMS
c                             ! which should cancel any pending
c                             ! programs in a command file.
      CALL EXIT
      END
####
```

```
C.............................................................
C
C     FUNCTION TO CALCULATE A SPECIFIED TIME
C
      FUNCTION TS(T01,DIST)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      TS = T01 + (DIST+RMAX)**(1./ALPHA1) /ALEPH
C
      RETURN
      END
####
```

```
C..............................................................
C
C     OBSERVER TRIAL AND ERROR FUNCTIONS
C       --- TUPF ---- TDNF ---
c
c     Modified 30 Jan 86 for more general trial and error scheme. ts
C
      FUNCTION TUPF(TOL)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      LOGICAL pflag         ! for diagnostic output
      pflag = .false.
C
      TMAX = RMAX**(1./ALPHA1)/ALEPH + TOL
      TMIN = TOL
      IF(TOL .LT. 0.) TMIN = 0.
c
c*** TMIN and TMAX represent the first and last time this observer could
c*** encounter the upwind edge of the source.  TMAX is the time when the
c*** observer passes over x=0, and TMIN is the time the observer is
c*** released (unless set to zero because the spill has not yet begun).
c*** Now, refine the guess of TMIN and TMAX by dividing the interval
c*** into 20 segments and checking if the observer crosses the upwind
c*** edge over the smaller interval starting with TMIN.
c
      DT = (TMAX - TMIN) / 20.
      TL = TMIN

      DO 10 I=1,19  ! I don't have to check the last interval.
      TL = TL + DT
      XO = XIT(TL, TOL)
      XG = -AFGEN(RADG, TL, 'tupf')
      DIF = XO - XG
      IF(DIF .LT. 0.) GOTO 10     ! observer has not yet reached upwind edge
            TMAX = TL      ! now observer has reached upwind edge
            TMIN = TL - DT
            DIFATMAX = DIF ! DIF at MAX
            GOTO 20
   10 CONTINUE
      TMIN = TMAX - DT
      DIFATMAX = DIF
  20  CONTINUE
c
c*** Now perform bisection search to get desired convergence between new TMAX
c*** and TMIN
c
      TL = (TMAX + TMIN)/2.
C
      DO 100 I = 1,20
      II = 0
 110  XG = -AFGEN(RADG,TL,'tupf')
      XO = XIT(TL,TOL)
      IF(XO .LT. 0.) GO TO 120
      TL = (TL+TMIN)/2.
      II = II + 1
      if(pflag) write(6,5020) tl,t01,xg,xo
5020  format(' tl:',1pg13.5,' t01:',1pg13.5,' xg:',1pg13.5,
     1 ' xo:',1pg13.5)
      IF(II.EQ. 20) GOTO 101       ! Kill the program.
      GO TO 110
C
 120  CONTINUE
      DIF = XO - XG
      sum = abs(xo + xg)/2. + ERTUPF
      IF(ABS(DIF)/sum .LT. ERTUPF) GO TO 1000
      if(pflag) write(6,5040) tmin,tmax,tl,xo,xg
5040  format(' tmin:',1pg13.5,' tmax:',1pg13.5,' tl:',1pg13.5,
     1 ' xo:',1pg13.5,' xg:',1pg13.5)
C
      IF( DIF*DIFATMAX .GT. 0.) THEN
            TMAX = TL        ! a new maximum for the range
            DIFATMAX = DIF   ! a new DIF for the maximum
      ELSE
            TMIN = TL        ! a new minimum for the range
      ENDIF
C
 100  TL = (TMAX + TMIN)/2.
C
c*** The above search scheme failed.  Before killing the program, check to
c*** see if the desired point falls on a transition from a blanket to a
c*** non-blanket situation.
c
      t1 = TL+.01
      T2 = TL-.01
      XG1 = AFGEN(RADG,T1,'tupf')
      xg2 = AFGEN(RADG,T2,'tupf')
      dif = abs(xg1-xg2)
      if(dif.gt. 100. .AND. (XO.GE.XG2 .AND. XO.LE.XG1)) then
            tupf = t2       ! jump from blanket to non-blanket occured
            RETURN
            ENDIF
```

```
c
c*** Kill the program.
c
 101  if(pflag) write(6,4000) RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND,alpha
 4000 format(' rm:',1pg13.5,' szm:',1pg13.5,' emax: ',1pg13.5,/,
     1 ' rmax:',1pg13.5,' tsc1:',1pg13.5,' aleph:',1pg13.5,/,
     2 ' tend:',1pg13.5,' alpha:',1pg13.5)
      if(pflag) write(6,4010) tmax,tmin
 4010 format(' tmax: ',1pg13.5,' tmin: ',1pg13.5)
            CALL trap(6)
C
c*** successful completion
c
 1000 TUPF = TL
      RETURN
      END
C
c
c
c
      FUNCTION TDNF(TOL)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      LOGICAL pflag
      pflag = .FALSE.
C
      TMIN = RMAX**(1./ALPHA1)/ALEPH + TOL
      IF(TMIN .LT. 0.) TMIN = 0.
      TMAX = (2.*RMAX)**(1./ALPHA1)/ALEPH + TOL
c
c*** TMIN and TMAX represent the first and last time this observer could
c*** encounter the downwind edge of the source.  TMAX is the time when the
c*** observer passes over x=+RMAX, and TMIN is the time the observer
c*** passes over x=0 (unless set to zero because the spill has not begun
c*** yet).  Now, refine the guess of TMIN and TMAX by dividing the interval
c*** into 20 segments and checking if the observer crosses the downwind
c*** edge over the smaller interval starting from TMAX.
c
      DT = (TMAX - TMIN) / 20.
      TL = TMAX
      if(pflag) write(6,*) 'tmax, tmin, dt: ',tmax, tmin, dt

      DO 10 I=1,19  ! I don't have to check the last interval.
      TL = TL - DT
      XO = XIT(TL, TOL)
      XG = AFGEN(RADG, TL, 'tupf')
      DIF = XO - XG
      if(pflag) write(6,*) 'tl, xo, xg, dif:',tl, xo, xg, dif
      IF(DIF .GT. 0.) GOTO 10      ! observer has passed the downwind edge
            TMIN = TL       ! now observer is about to reach downwind edge
            TMAX = TL + DT
            GOTO 20
   10 CONTINUE
c
      TMAX = TMIN + DT
   20 CONTINUE
      XO = XIT(TMAX, TOL)
      XG = AFGEN(RADG, TMAX, 'tupf')
      DIFATMAX = XO - XG
      if(pflag) then
            write(6,*) '20: tmax, tmin: ',tmax, tmin
            write(6,*) 'difatmax: ',difatmax
      endif
C
c*** Now perform bisection search to get desired convergence between new TMAX
c*** and TMIN.
c
      TL = (TMAX + TMIN)/2.
C
      DO 100 I = 1,20
      II = 0
  110 XG = AFGEN(RADG, TL, 'tdnf')
      XO = XIT(TL,TOL)
C
      IF(XO .GT. 0.) GO TO 120
      TL = (TMAX + TL)/2.
      II = II + 1
      if(pflag) write(6,5020) tl,t01,xg,xo
 5020 format(' tl:',1pg13.5,' t01:',1pg13.5,' xg:',1pg13.5,
     1 ' xo:',1pg13.5)
      IF(II.EQ. 20) GOTO 101       ! Kill the program.
      GO TO 110
C
  120 CONTINUE
C
      DIF = XO - XG
      sum = abs(xo+xg)/2. + ERTDNF
      IF(ABS(DIF)/sum .LT. ERTDNF) GO TO 1000
      if(pflag) write(6,5040) tmin,tmax,tl,xo,xg
 5040 format(' tmin:',1pg13.5,' tmax:',1pg13.5,' tl:',1pg13.5,
      1 ' xo:',1pg13.5,' xg:',1pg13.5)
C
      IF( DIF*DIFATMAX .GT. 0.) THEN
            TMAX = TL       ! a new maximum for the range
            DIFATMAX = DIF  ! a new DIF for the maximum
      ELSE
            TMIN = TL       ! a new minimum for the range
      ENDIF
C
  100 TL = (TMAX + TMIN)/2.
C
c*** The above search scheme failed.  Before killing the program, check to
c*** see if the desired point falls on a transition from a blanket to a
c*** non-blanket situation.
c
      t1 = TL+.01
      T2 = TL-.01
      XG1 = AFGEN(RADG,T1,'tupf')
      xg2 = AFGEN(RADG,T2,'tupf')
      dif = abs(xg1-xg2)
      if(dif.gt. 100. .AND. (XO.LE.XG2 .AND. XO.GE.XG1)) then
            tdnf = t2      ! Jump from blanket to non-blanket occured
            RETURN
            ENDIF
c
c*** Kill the program.
c
  101 if(pflag) write(6,4000) RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND,alpha
 4000 format(' rm:',1pg13.5,' szm:',1pg13.5,' emax: ',1pg13.5,/,
      1 ' rmax:',1pg13.5,' tsc1:',1pg13.5,' aleph:',1pg13.5,/,
      2 ' tend:',1pg13.5,' alpha:',1pg13.5)
      if(pflag) write(6,4010) tmax,tmin
 4010 format(' tmax: ',1pg13.5,' tmin: ',1pg13.5)
            CALL trap(7)
C
c*** successful completion
c
 1000 TDNF = TL
      RETURN
      END
####
```

```
C..........................................................
C
C     FUNCTIONS ASSOCIATED WITH THE OBSERVER CALCULATIONS
C
C..........................................................
C
C     FUNCTION TO RETURN OBSERVER VELOCITY AS A FUNCTION OF TIME
C
      FUNCTION UIT(T,T01)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      UIT = ALPHA1 * ALEPH**ALPHA1 *(T-T01)**ALPHA
C
      RETURN
      END
c
c
C..............................................................
C
C     FUNCTION TO RETURN POSITION AS A FUNCTION OF TIME AND TO
C
C
      FUNCTION XIT(T1,T01)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      xit = -rmax
      arg = t1-t01
      if(arg .le. 0.) return
      XIT = (ALEPH*(T1 - T01))**ALPHA1 - RMAX
C
      RETURN
      END
c
c
c................................................................
c
C
C     FUNC TO RETURN A VALUE OF TO BASED ON A POSITION AND TIME
C
```

```
      FUNCTION TOOB(X,T)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      ARG = 0.
      CHECK = ABS((ABS(X)-ABS(RMAX)))/(ABS(X)+ABS(RMAX))
      IF(CHECK .GT. 0.001) ARG = (X + RMAX)**(1./ALPHA1)/ALEPH
      TOOB = T - ARG
      RETURN
      END
####
```

APPENDIX G

INTERFACE PROGRAM CODE LISTING
(OOMS_IN and DEGBRIDGE)

Preceding page blank

```
      Program ooms_in
c
c
c     OOMS_IN is designed to perform two tasks including:
c
c     a) Read the input file for the OOMS/DEGADIS model input.  From
c     this information, OOMS_IN prepares the necessary file to run the
c     OOMS model.  The information necessary for DEGADIS is also included
c     in the single input file to minimize the user effort required.
c
c     b) Generate a command file to execute the OOMS/DEGADIS model from
C     the input information.  This portion is highly dependent on the
c     VAX/VMS environment because of the manipulation of character strings
c     and the commands necessary.
c
c...................................................................
c
c     This program can be invoked interactively or in a command file
c     submitted to batch.  For either case, the syntax is:
c
c     $ RUN SYS$DEGADIS:OOMS_IN
c     RUN_NAME
c
c     where RUN_NAME is the file name given to the particular simulation.
c     The input file RUN_NAME.IN is to be created before this program
c     is invoked using free formats.  The order of input is as follows:
c
c     <TITLE1>
C     <TITLE2>
C     <TITLE3>
C     <TITLE4>
C
C     <UO>  <ZO>
C     <ZR>
C     <INDVEL>  <ISTAB>  <MOLEN>
C     <TAMB>  <PAMB>  <RELHUM>
C     <TSURF>
C
C     <GASNAM>
C     <GASMW>
C     <AVTIM>
C     <JETTEM>
C     <GASUL>  <GASLL>  <ZLL>
C     <INDHT>  <CPC>  <CPP>
C
C     <ERATE>
C     <JETELE>  <JETDIA>
C     <TEND>
c     <XO>  <H>  <DISTA>  <DISTAN>
C     <K1>  <K2>  <K3>
C
```

```
C     Note that for readability, blank lines were inserted between the
c     input sections specifying a simulation title, atmospheric conditions,
c     gas properties, and the particular release conditions.  Symbol
c     definitions are as follows:
c
c     <TITLE1>, <TITLE2>, <TITLE3>, and <TITLE4> are four lines of up
c           to 80 characters each of a title for this simulation.
c
c     <U0> (m/s) is the ambient wind velocity at <Z0> (m).
c
c     <ZR> is the surface roughness (m).
c
c     <INDVEL> is an indicator which determines the method of calculation
c           for the ambient velocity profile in Ooms' model as follows:
c
c           For <INDVEL>=1, the Pasquill-Gifford stability category
c              (in <ISTAB> using 1 for A, 2 for B, etc.) is used
c              along with <ZR> to determine the Monin-Obukhov length
c              <MOLEN>; the log velocity profile is then determined
c              using <MOLEN>.
c
c           For <INDVEL>=2, the Monin-Obukhov length <MOLEN> is supplied
c              by the user; the log velocity profile is then determined
c              using <MOLEN>.
c
c     <TAMB>, <PAMB>, and <RELHUM> are the ambient temperature (K), the
c           ambient pressure (atm or N/m**2), and the relative humidity (%),
c           respectively.
c
c     <TSURF> is the surface temperature (K); if <TSURF> < 250 K, <TSURF> is
c           set to <TAMB>.
c
c     <GASNAM> is a three-letter designation for the contaminant's name.
c
c     <GASMW> is the contaminant's molecular weight (kg/kmole).
c
c     <AVTIM> is the averaging time (s).  At present, this parameter is
c           used to estimate the value of <DELTA> in DEGADIS.
c
c     <JETTEM> is the temperature of the jet (K).
c
c     <GASUL> and <GASLL> are the concentrations to be used for estimating
c           contours for an upper and lower concentration level in
c           DEGADIS.  The calculations are made for the elevation
c           <ZLL>.  Note that the DEGADIS computations will be carried
c           out to <GASLL>/2.
c
c     <INDHT> is used to include heat transfer in the DEGADIS computations.
c           Heat transfer is not included for <INDHT>=0.  For <INDHT>=1,
c           heat transfer is included, and the heat transfer coefficient
c           is calculated by DEGADIS.  <CPP> and <CPC> are used to
```

```
c           calculate the heat capacity in DEGADIS as a function of
c           temperature according to the correlation included in
c           DEGADIS.  (If this function is used, the average gas
c           temperature is used to estimate the mean (constant) heat
c           capacity in Ooms' model.)  If a constant heat capacity is
c           desired, set <CPP> to 1.0 and <CPC> to the desired heat
c           capacity (J/kg K).
c
c     <ERATE> is the mass evolution rate (kg/s).
c
c     <JETELE> is the inital jet elevation (m), and <JETDIA> is the initial
c           jet diameter (m).
c
c     <TEND> is the duration of the release.  For steady-state releases,
c           set <TEND> to 0.; to run the Ooms model only, set
c           <TEND> to a negative number.
c
c     <X0> is the starting value of jet trajectory integration (m).  <H>
c           is the step size for Ooms' model (m).  <DISTA> is the
c           downwind distance to the first output value (m).  <DISTAN> is
c           the distance between output points (m).
c
c     <K1>, <K2>, and <K3> determine the orientation of the jet with
c           respect to the ambient wind velocity as follows:
c
c           <K1> is for horizontal jets directed upwind (-1) and
c              downwind (+1); <K1>=0 for other directions.
c
c           <K2> is for vertical jets directed upward (+1) and
c              downward (-1); <K2>=0 for other directions.
c
c           <K3> is for horizontal jets directed to the left (-1) and
c              to the right (+1); <K3>=0 for other directions.
c
c           Note that only one of <K1>, <K2>, or <K3> can be nonzero
c           at any time.
c
c
c     T. Spicer
c     University of Arkansas
c     Department of Chemical Engineering
c     Fayetteville, AR 72701
c
c     (501) 575-4951
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      real mwa, mwg, molen, jettem,jetele,jetdia
C
```

```
      logical check4
c
      character*80 TITLE(4)
C
      character*3 gas_name
C
      character*100 OPNRUP
      character OPNRUP1(100)
      equivalence (opnrup1(1),opnrup)
      character*4 IN,er1,er2,er3,com,scl,sr3,lis
      character*4 dummy
      character*3 plus, gasnam
      character*2 con
      DATA POUND/'//  '/,POUNDN/-1.E-20/
C
      DATA IN/'.IN '/,er1/'.er1'/,er2/'.er2'/,er3/'.er3'/
      data scl/'.scl'/,sr3/'.sr3'/,lis/'.lis'/
      data com/'.com'/
      data plus/' + '/,con/' -'/
c
C*** GET THE FILE NAME TO BE USED BY ALL OF THE ROUTINES
C
      READ(5,820) NCHAR,opnrup
      opnrup = opnrup(1:nchar) // in(1:4)
C
C*** Now get the rest of the desired information from RUN_NAME.IN
C
      open(unit=8,name=opnrup,type='old')

      read(8,8000) title(1)
      read(8,8000) title(2)
      read(8,8000) title(3)
      read(8,8000) title(4)
 8000 format(a80)
 8010 format(a3)
      read(8,*) u0, z0
      read(8,*) zr

      read(8,*) indvel, istab, molen
      if( indvel.ne.1 .or. indvel.ne.2) indvel = 1
      if( istab.le.0 .or. istab.gt.6) istab = 4
      if(indvel .eq. 1) then
            if(istab.eq.1) then
             molen = -11.43 * zr**0.103
            else if(istab.eq.2) then
             molen = -25.98 * zr**0.171
            else if(istab.eq.3) then
             molen = -123.4 * zr**0.304
            else if(istab.eq.4) then
             molen = 0.0
            else if(istab.eq.5) then
```

```
             molen = 123.4 * zr**0.304
            else if(istab.eq.6) then
             molen = 25.98 * zr**0.171
            endif
      endif
      vkcon = 0.35
      ustar = u0*vkcon/(dlog((z0+zr)/zr) - psif(z0,molen))

      read(8,*) tamb, pamb, relhum
      if( tamb.le.0. ) tamb = tamb+273.15D0
      if( pamb.gt.1.1 ) pamb = pamb/101325.D0
      if( relhum.lt.0. .or. relhum.gt.100. ) relhum = 50.

      read(8,*) tsurf
      if( tsurf.lt.250. ) tsurf = tamb

      read(8,8010) gasnam
      read(8,*) gasmw
      read(8,*) avtim
      read(8,*) jettem
      read(8,*) gasul, gasll, zll
      if( gasll.le.0. ) gasll = 0.01
      if( gasul.le.gasll ) gasul = dmax1( 1.1D0*gasll, 1.0D0)

      read(8,*) indht, cpc, cpp
      read(8,*) erate
      read(8,*) jetele, jetdia
      read(8,*) tend
      check4 = .true.
      if(tend .gt. 0.) check4 = .false.

      read(8,*) x0, h, dista, distan
      read(8,*) k1, k2, k3
      if( k1.eq.0 .and. k2.eq.0 .and. k3.eq.0 ) then
            k2 = +1        ! vertical upward
      else if( k1.ne.0 ) then
            k2 = 0
            k3 = 0
      else if( k2.ne.0 ) then
            k1 = 0
            k3 = 0
      else if( k3.ne.0 ) then
            k1 = 0
            k2 = 0
      endif
c
c*** It is time to prepare the input file for the OOMS model.
c
      opnrup = opnrup(1:nchar) // '.ino'
      open(unit=1,name=opnrup,type='new')
      write(1,*) '              1'
      write(1,603) title(1)

      write(1,600) x,h,dista,distan

      ta = tamb
      ts = jettem
      ur = ustar/vkcon*(dlog((JETELE+zr)/zr) - psif(JETELE,molen))
      write(1,601) ta, ts, ur, z0, jetele

      ust = ustar
      write(1,611) ust, zr, molen, istab

      mwa = 28.96D0
      mwg = gasmw
      cpa = 1006.3
      if( cpp .eq. 1.) then
            cpg = cpc
      else
            tmean = (tamb+jettem)/2.
            cpg = (3.33D4 + cpc*(tmean**cpp - jettem**cpp)/
     .                (tmean - jettem))/gasmw
      endif
      write(1,601) mwa, mwg, cpa, cpg

      rhoe = pamb*101325.D0*gasmw/8314.D0/jettem
      diajet = jetdia*100.D0
      qinit = erate/rhoe
      write(1,604) diajet, qinit

      dc = 1000.
      aloc = 0.
      bloc = dc - jetele
      write(1,606) dc, aloc, bloc

      write(1,609) k1, k2, k3

      alfa1 = 0.057
      alfa2 = 0.5
      alfa3 = 1.0
      write(1,610) alfa1, alfa2, alfa3

  600 format(4(5x, f10.3))
  601 format(5(5x, f10.3))
  603 format(a80)
  604 format(5x, f10.3, 5x, e10.3, 5x, f10.3)
  606 format(3(5x, f10.3))
  609 format(3(5x, i10))
  610 format(3(5x, f10.3), 5x, f10.4)
  611 format(5x, f10.3, 2(5x, e10.4), 5x, i10)
      close(unit=1)
```

```
c
c*** Now, prepare the command file
c
C
C*** FORMATS
C
  820 FORMAT(Q,A40)
c
c
 1210 format(a4)
      opnrup = opnrup(1:nchar) // com(1:4)
c
      open(unit=8,name=opnrup,type='new',
     $ carriagecontrol='list',recordtype='variable')
c
c*** Lines to start the Ooms model and invoke DEGBRIDGE
c
      opnrup = opnrup(1:nchar) // '.ino for001'
      write(8,1100) (opnrup1(i),i=1,nchar+11)
 1100 format('$ assign ',51a1)
      opnrup = opnrup(1:nchar) // '.out for003'
      write(8,1100) (opnrup1(i),i=1,nchar+11)
      opnrup = opnrup(1:nchar) // '.ind for002'
      write(8,1100) (opnrup1(i),i=1,nchar+11)
      write(8,1160)
 1160 format('$ run sys$degadis:ooms')
      write(8,1170)
 1170 format('$ deassign for001',/,'$ deassign for002',/,
     .      '$ deassign for003')

      if(tend .lt. 0.) goto 3000
      write(8,1180)
 1180 format('$ run sys$degadis:degbridge')
      write(8,1290) (opnrup1(i),i=1,nchar)
c
c
c
c
      opnrup = opnrup(1:nchar) // er1(1:4)
c
      write(8,1250) (opnrup1(i),i=1,nchar+4)
 1250 format('$ copy/log SYS$DEGADIS:example.er1 ',40a1)
      IF(u0 .eq. 0.) then
            write(8,1280)
            write(8,1290) (opnrup1(i),i=1,nchar)
            goto 1340
            endif
      opnrup = opnrup(1:nchar) // er2(1:4)
c
      write(8,1260) (opnrup1(i),i=1,nchar+4)
 1260 format('$ copy/log SYS$DEGADIS:example.er2 ',40a1)
      opnrup = opnrup(1:nchar) // er3(1:4)
c
      if(.not.check4) then  ! transient
c
            write(8,1270) (opnrup1(i),i=1,nchar+4)
 1270       format('$ copy/log SYS$DEGADIS:example.er3 ',40a1)
c
            write(8,1280)
 1280       format('$ run SYS$DEGADIS:DEGADIS1')
            write(8,1290) (opnrup1(i),i=1,nchar)
 1290       format(40a1)
            write(8,1300)
 1300       format('$ run SYS$DEGADIS:DEGADIS2')
            write(8,1290) (opnrup1(i),i=1,nchar)
            write(8,1320)
 1320       format('$ run SYS$DEGADIS:DEGADIS3')
            write(8,1290) (opnrup1(i),i=1,nchar)
c
      else
            write(8,1280)
            write(8,1290) (opnrup1(i),i=1,nchar)
c
            write(8,1330)
 1330       format('$ run SYS$DEGADIS:SDEGADIS2')
            write(8,1290) (opnrup1(i),i=1,nchar)
c
      endif

      opnrup = opnrup(1:nchar) // '.out' //
     .       plus(1:3) //opnrup(1:nchar) // scl(1:4) //
     .       plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
      write(8,1370) (opnrup1(i),i=1,3*nchar+20)
 1370 format('$ copy/log ',100a1)
      opnrup = opnrup(1:nchar) // lis(1:4)
      write(8,1390) (opnrup1(i),i=1,nchar+4)
 1390 format('  ',40a1)
c
 1340 close(unit=8)

 3000 continue
      write(6,2099)
 2099 format(/,' OOMS_IN - beginning command file.')
      opnrup = '@' // opnrup(1:nchar) // ' '
      istat = lib$do_command(opnrup)
      write(6,2100)
 2100 format(/,' ?OOMS_IN? command file failed to start.')
c
      CALL EXIT
      END
####

 8 -- sys$degadis:ooms_in.for                   16-NOV-1987 06:28:11
```

```
      Program DEGBRIDGE
c
c
c.....................................................................
c
c
c     DEGBRIDGE is designed read the input file for the
c     OOMS/DEGADIS model input.  From this information and the
c     output from Ooms' model, DEGBRIDGE prepares the necessary
c     file to run DEGADIS (RUN_NAME.INP).
c
c.......................................................................
c
c     The structure of the RUN_NAME.IN file is outlined in OOMS_IN.
c
c     The output file supplied from Ooms' model contains four (4)
c     variables as follows (in RUN_NAME.IND):
c
c     <DIST>  <CONCL>  <TEMCL>  <BOOMS>
c
c     where <DIST> is the distance to the centerline touchdown (m),
c     <CONCL> is the centerline concentration (kg/m**3) at <DIST>,
c     <TEMCL> is the centerline temperature (K) at <DIST>, and
c     <BOOMS> is the value of Ooms' b (m) at <DIST>.
c
c     To establish the initial conditions for DEGADIS, the source
c     concentration is assumed to be <CONCL>, and the source
c     radius is sqrt(2.)*<BOOMS> which is consistent with Ooms'
c     development.
c
c
c     T. Spicer
c     University of Arkansas
c     Department of Chemical Engineering
c     Fayetteville, AR 72701
c
c     (501) 575-4951
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
c
      include 'sys$degadis:DEGADISIN.dec'
c
      COMMON
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
C
C
       real mwa, mwg, molen, jettem,jetele,jetdia
C
       REAL*8 ML,K
       LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
       character*80 TITLE(4)
       character*3 gas_name
       character*4 pound
       character*24 TSRC,TINP,TOBS,TSRT
C
       character*100 OPNRUP
       character OPNRUP1(100)
       equivalence (opnrup1(1),opnrup)
       character*4 IN,er1,er2,er3,com,scl,sr3,lis
       character*4 dummy
       character*3 plus, gasnam
       character*2 con
c
       data wmw/18.02D0/
       data wma/28.96D0/
c
       DATA POUND/'//  '/,POUNDN/-1.D-20/
C
       DATA IN/'.IN '/,er1/'.er1'/,er2/'.er2'/,er3/'.er3'/
       data scl/'.scl'/,sr3/'.sr3'/,lis/'.lis'/
       data com/'.com'/
       data plus/' + '/,con/' -'/
c
C*** GET THE FILE NAME
C
       write(6,*) 'Beginning DEGBRIDGE'
       READ(5,820) NCHAR,opnrup
       opnrup = opnrup(1:nchar) // in(1:4)
C
C*** Now get the rest of the desired information from RUN_NAME.IN
C
       open(unit=8,name=opnrup,type='old')

       read(8,8000) title(1)
       read(8,8000) title(2)
       read(8,8000) title(3)
       read(8,8000) title(4)
 8000  format(a80)
 8010  format(a3)
```

```
      read(8,*) u0, z0
      read(8,*) zr

      read(8,*) indvel, istab, molen
      if( indvel.ne.1 .or. indvel.ne.2) indvel = 1
      if( istab.le.0 .or. istab.gt.6) istab = 4
      if(indvel .eq. 1) then
            if(istab.eq.1) then
             molen = -11.43 * zr**0.103
            else if(istab.eq.2) then
             molen = -25.98 * zr**0.171
            else if(istab.eq.3) then
             molen = -123.4 * zr**0.304
            else if(istab.eq.4) then
             molen = 0.0
            else if(istab.eq.5) then
             molen = 123.4 * zr**0.304
            else if(istab.eq.6) then
             molen = 25.98 * zr**0.171
            endif
      endif
      vkcon = 0.35
      ustar = u0*vkcon/(dlog((z0+zr)/zr) - psif(z0,molen))

      read(8,*) tamb, pamb, relhum
      if( tamb.le.0. ) tamb = tamb+273.15D0
      if( pamb.gt.1.1 ) pamb = pamb/101325.D0
      if( relhum.lt.0. .or. relhum.gt.100. ) relhum = 50.

      vaporp = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
      sat = 0.622*vaporp / (pamb- vaporp)
      humid = relhum/100. * sat

      read(8,*) tsurf
      if( tsurf.lt.250. ) tsurf = tamb

      read(8,8010) gasnam
      read(8,*) gasmw
      read(8,*) avtim
      read(8,*) jettem
      read(8,*) gasul, gasll, zll
      if( gasll.le.0. ) gasll = 0.01
      if( gasul.le.gasll ) gasul = dmax1( 1.1D0*gasll, 1.0D0)

      read(8,*) indht, cpc, cpp
      if(indht.ne.0 .or. indht.ne.1 .or. indht.ne.2) indht=1
      if(cpp .eq. 1.D0) cpc = cpc*gasmw - 3.33D4

      read(8,*) erate
      read(8,*) jetele, jetdia
      read(8,*) tend
      check4 = .true.
      if(tend .gt. 0.) check4 = .false.

      read(8,*) x0, h, dista, distan
      read(8,*) k1, k2, k3
      if( k1.eq.0 .and. k2.eq.0 .and. k3.eq.0 ) then
            k2 = +1         ! vertical upward
      else if( k1.ne.0 ) then
            k2 = 0
            k3 = 0
      else if( k2.ne.0 ) then
            k1 = 0
            k3 = 0
      else if( k3.ne.0 ) then
            k1 = 0
            k2 = 0
      endif
c
      close(unit=8)
C
C*** Now get the rest of the desired information from RUN_NAME.IND
C
      opnrup = opnrup(1:nchar) // '.ind'
      open(unit=8,name=opnrup,type='old')
      read(8,*) odist, ocon, otemp, ob
      close(unit=8)

      yc = ocon*8314.D0*otemp/101325.D0/pamb/gasmw
      ya = (1.D0-yc)/(1.D0+humid*wma/wmw)
      yw = 1.D0-yc-ya
      wm = yc*gasmw + ya*wma + yw*wmw
      wc = gasmw/wm * yc
      rho = 101325.D0*pamb*wm/8314.D0/otemp
c
c*** It is time to prepare the input file for DEGADIS
c
      opnrup = opnrup(1:nchar) // '.inp'
      OPEN(UNIT=8,NAME=OPNRUP,TYPE='NEW',
     $  carriagecontrol='list',recordtype='variable')
C
      DO 100 I=1,4
 100  WRITE(8,1135) TITLE(I)
C
c*** Atmospheric parameters:
c
      WRITE(8,1020) U0,Z0,ZR
      WRITE(8,1040) istab
C
      goto(161,162,163,164,165,166) istab
 161  delta = 0.224         ! A
      beta = 0.894
```

```
        sisx_coeff = 0.02
        sisx_pow = 1.22
        sisx_min_dist = 130.
        goto 170
  162   delta = 0.164          ! B
        beta = 0.894
        sisx_coeff = 0.02
        sisx_pow = 1.22
        sisx_min_dist = 130.
        goto 170
  163   delta = 0.109          ! C
        beta = 0.894
        sisx_coeff = 0.02
        sisx_pow = 1.22
        sisx_min_dist = 130.
        goto 170
  164   delta = 0.071          ! D
        beta = 0.894
        sisx_coeff = 0.04
        sisx_pow = 1.14
        sisx_min_dist = 100.
        goto 170
  165   delta = 0.053          ! E
        beta = 0.894
        sisx_coeff = 0.17
        sisx_pow = 0.97
        sisx_min_dist = 50.
        goto 170
  166   delta = 0.036          ! F
        beta = 0.894
        sisx_coeff = 0.17
        sisx_pow = 0.97
        sisx_min_dist = 50.
  170   continue

        ml = molen
        WRITE(8,1020) DELTA,BETA,ml
        WRITE(8,1020) sisx_coeff,sisx_pow,sisx_min_dist

c
c*** ambient pressure, temperatures, and humidity
c

        write(8,1025) tamb,pamb,humid
c
        isofl = 0
        ihtfl = indht
        htco  = 0.
        iwtfl = 0
        wtco  = 0.
c
```

```
      write(8,1060) isofl,tsurf
      write(8,1060) ihtfl,htco
      write(8,1060) iwtfl,wtco
c
C
c*** gas characteristics
c
      gas_name = gasnam
      write(8,1415) gas_name
c
      gas_mw = gasmw
      gas_temp = jettem
      gas_rhoe = 101325.D0*pamb/8314.D0*gasmw/jettem
      gas_cpk = cpc
      gas_cpp = cpp
      gas_ufl= gasul
      gas_lfl= gasll
      gas_zsp= zll

      WRITE(8,1020) gas_mw, gas_temp,gas_rhoe
      write(8,1020) gas_cpk,gas_cpp
      WRITE(8,1020) gas_ufl,gas_lfl,gas_zsp
c
      cclow = (gas_lfl/2.) * gas_rhoe * (gas_temp/tamb)
      WRITE(8,1010) CcLOW
c
c
c*** source description FOR A DILUTED SOURCE ...
C
      gmass0 = 0.          ! no initial cloud mass
      write(8,1020) gmass0
      np = 4
c
      if(check4) tend = 6023. ! [=] sec
c
      ess = erate
      r1ss = 1.414213 * ob  ! sqrt(2.)*ob
      pwc(1) = wc
      ptemp(1) = otemp
      pfracv(1) = 1.0
c
      PTIME(1) = 0.
      et(1) = ess
      r1t(1)= r1ss
      PWC(1)= pwc(1)
      PTEMP(1)= ptemp(1)
      PFRACV(1)= 1.0D0
      PTIME(2) = tend
      et(2) = ess
      r1t(2)= r1ss
      PWC(2)= pwc(1)
      PTEMP(2)=ptemp(1)
      PFRACV(2)= 1.0D0
      PTIME(3) = tend + 1.
      et(3) = 0.
      r1t(3)= 0.
      PWC(3)= pwc(1)
      PTEMP(3)=ptemp(1)
      PFRACV(3)= 1.0D0
      PTIME(4) = tend + 2.
      et(4) = 0.
      r1t(4)= 0.
      PWC(4)= pwc(1)
      PTEMP(4)=ptemp(1)
      PFRACV(4)= 1.0D0
      slen = 2.D0*r1ss
      swid = pi*r1ss**2/slen/2.D0

      WRITE(8,1040) NP
      DO 800 I=1,NP
  800 WRITE(8,1030) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),PFRACV(I)
C
      write(8,*) check1,check2,again,check3,check4,check5
C
      istat = lib$date_time(tinp)
      write(8,1070) tinp
c
      if(check4) write(8,1020) ess,slen,swid        ! steady state
c
C
      CLOSE(UNIT=8)
C
C
C*** FORMATS
C
  820 FORMAT(Q,A40)
c
 1010 format(1x,1pg14.7)
 1020 format(3(1x,1pg14.7))
 1025 format(5(1x,1pg14.7))
 1030 format(1x,5(1pg14.7,1x),1pg14.7)
 1040 format(1x,I5)
 1050 format(a24)
 1060 format(1x,i4,1x,1pg14.7)
 1070 format(a24)
c
 1134 FORMAT(A80)
 1135 FORMAT(A80)
c
c
c
 1415 format(a3)
```

```
C
C
C
      CALL EXIT
      END
****
```

APPENDIX H
PARTIAL LISTING OF PROGRAM VARIABLES

Variable	Data Type	Symbol	Units	Comments
AGAIN	LOGICAL			Local communications in SSSUP
ALEPH	REAL			Collection of constants to calculate observer position and velocity
ALPHA	REAL	α	n/a	Power law velocity profile power
ALPHA1	REAL	$(1.0 + \alpha)$	n/a	
BETA	REAL	β	n/a	Lateral similarity power
CCLOW	REAL		kg/m^3	Lowest mixture concentration of interest
CHECK1	LOGICAL			Unused logical flag
CHECK2	LOGICAL			When true, release type without a liquid source
CHECK3	LOGICAL			Local communications flag used in DEGADIS1
CHECK4	LOGICAL			When true, steady-state simulation
CHECK5	LOGICAL			When true, user sets time-sort parameters
DELTA	REAL	δ	$m^{1-\beta}$	Lateral similarity coefficient
DEN(1,I)	REAL	y_c	mole fraction	Contaminant mole fraction
DEN(2,I)	REAL	c_c	kg/m^3	Contaminant concentration for the given mole fraction

Variable	Data Type	Symbol	Units	Comments
DEN(3,I)	REAL	ρ	kg/m^3	Mixture density for the given mole fraction
DEN(4,I)	REAL	h	J/kg	Mixture enthalpy for the given mole fraction
DEN(5,I)	REAL	T	K	Mixture temperature for the given mole fraction
EMAX	REAL		kg/s	Maximum of secondary source mass evolution rate
ESS	REAL	E	kg/s	Steady-state release rate
ET(I)	REAL	E(t)	kg/s	Source mass evolution rate as a function of time PTIME(I)
G	REAL	g	m/s^2	Acceleration due to gravity
GAMMAF	REAL	$\Gamma(1/(1+\alpha))$	n/a	
GAS_CPK	REAL	q_1	J/kmol K$^{p_1-1}$	Constant for contaminant heat capacity
GAS_CPP	REAL	p_1	n/a	Power for contaminant heat capacity
GAS_LFL	REAL		mole fraction	Lower contaminant concentration level for estimating contours
GAS_MW	REAL	MW_c	kg/kmol	Contaminant molecular weight
GAS_NAME	CHARACTER*3			Name of contaminant
GAS_RHOE	REAL	ρ_0	kg/m^3	Saturated vapor density of contaminant at T_0
GAS_TEMP	REAL	T_0	K	Contaminant storage temperature
GAS_UFL	REAL		mole fraction	Upper contaminant concentration level for estimating contours

Variable	Data Type	Symbol	Units	Comments
GAS_ZSP	REAL		m	Height for estimating contours
GMASS0	REAL		kg	Initial mass of gas over the primary source
HTCO	REAL	h_0 V_H	J/m^2sK m/s	Constant coefficient when IHTFL=-1 LLNL heat transfer velocity when IHTFL=2
HUMID	REAL		kg water/ kg dry air	Ambient absolute humidity
IHTFL	INTEGER			Heat transfer flag: IHTFL=-1 constant coefficient IHTFL=0 no heat transfer IHTFL=1 DEGADIS correlation IHTFL=2 LLNL correlation
ISOFL	INTEGER			Isothermal release when ISOFL=1
ISTAB	INTEGER			Pasquill atmospheric stability indicator (ISTAB=1 for A, (ISTAB=2 for B, etc.)
IWTFL	INTEGER			Water transfer flag IWTFL=-1 constant coefficient IWTFL=0 no water transfer IWTFL=1 DEGADIS correlation
K	REAL	k	n/a	von Karman's constant, 0.35
LUNLOG	INTEGER			Fortran logical unit number which acts as a simulation log
MAXNOB	INTEGER			Maximum number of observers
ML	REAL	λ	m	Monin-Obukhov length
NOBS	INTEGER			Number of observers for the pseudosteady-state simulation

Variable	Data Type	Symbol	Units	Comments
NREC(I,1)	INTEGER			Number of records generated in PSSOUT for observer I
NREC(I,2)	INTEGER			Number of records generated in SSGOUT for observer I
PAMB	REAL	p	atm	Ambient pressure
POUND	CHARACTER*4			Character string to signal end of data ('// ')
POUNDN	REAL			Numerical value to signal end of data (-1.E-20)
QSTR(1,I)	REAL	t	s	Independent variable time for ordered pairs QSTR
QSTR(2,I)	REAL	Q_*	kg/m^2s	Atmospheric takeup rate as a function of time
RADG(1,I)	REAL	t	s	Independent variable time for ordered pairs RADG
RADG(2,I)	REAL	R	m	Secondary source radius as a function of time
RELHUMID	REAL		%	Ambient relative humidity
RHOA	REAL	ρ_a	kg/m^3	Ambient air density
RM	REAL	R_m	m	Radius at EMAX (when secondary source mass evolution rate is a maximum)
RMAX	REAL	R_{max}	m	Maximum secondary source radius
RT2	REAL	$\sqrt{2}.$	n/a	Constant
R1SS	REAL	R_p	m	Steady-state primary source radius

Variable	Data Type	Symbol	Units	Comments
R1T(I)	REAL	R_p	m	Primary source radius as a function of time PTIME(I)
SIGX_COEFF	REAL			Along-wind similarity coefficient
SIG_MIN_DIST	REAL		m	Minimum distance to apply x-direction dispersion correction
SIGX_POW	REAL		n/a	Along-wind similarity power
SLEN	REAL	L	m	Steady-state source length
SQPIO2	REAL	$\sqrt{\pi/2.}$	n/a	Constant
SQRTPI	REAL	$\sqrt{\pi}$	n/a	Constant
SRCDEN(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCDEN
SRCDEN(2,I)	REAL	ρ	kg/m^3	Secondary source density as a function of time
SRCENTH(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCENTH
SRCENTH(2,I)	REAL	h	J/kg	Secondary source enthalpy as a function of time
SRCWA(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCWA
SRCWA(2,I)	REAL	w_a	mass fraction	Secondary source air mass fraction as a function of time
SRCWC(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCWC

Variable	Data Type	Symbol	Units	Comments
SRCWC(2,I)	REAL	w_c	mass fraction	Secondary source contaminant mass fraction as a function of time
SWID	REAL		m	Steady-state source half-width
SZM	REAL	S_{z0m}	m	Value of S_{z0} at EMAX (when secondary source mass evolution rate is a maximum)
[illegible]	REAL	T	K	Ambient temperature
TEND	REAL		s	Termination time of secondary source
TINP	CHARACTER*24			Time DEGADISIN was executed
TITLE(1:4)	CHARACTER*80			Text title block 4 lines of 80 spaces
TO(I)	REAL		s	Time of release for observer I
TSURF	REAL	T_s	K	Surface temperature
USTAR	REAL	u_*	m/s	Friction velocity
U0	REAL	u_0	m/s	Ambient velocity at height z_0
WTCO	REAL	F	kg/m^2s	Mass transfer coefficient when IWTFL=-1
XV(I)	REAL	x_v	m	Virtual source position for estimation of S_y in SSG
Z0	REAL	z_0	m	Height for velocity u_0
ZR	REAL	z_R	m	Roughness length

APPENDIX I

DEGADIS DIAGNOSTIC MESSAGES

To assist the user in determining the source of any problems, a diagnostic procedure has been included in DEGADIS. The subroutine TRAP is meant to cause an orderly termination of the program for many detected errors. It performs two basic functions: TRAP displays an error code and a single line diagnostic message giving the reason for premature termination, and TRAP forces an output of the COMMON area data sets to the file TRAP.DBG.

The first three lines sent to the execution log (default=TERMINAL) include the TRAP introductory lines and the error code number:

```
The best laid plans of mice and men . . .
You have entered a TRAP--THE LAND OF NO RETURN
CODE:  NN
```

where NN represents the code of the error message which follows in the log. The error message begins with the name of the calling routine.

The following is a list of the error codes, error messages, and suggested actions for each problem.

Code: 1

```
DEGADIS1?  Source integration has returned IHLF=NN
```

Action: This error occurs during integration of the equations which describe the gas source. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (STPIN in the ER1 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERBND in the ER1 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (one of the following: WTRG, WTTM, WTYA, WTYC, WTEB, WTMB, or WTUH in the ER1 file).

When IHLF=12, the initial increment of the independent variable (STPIN) is 0. Correct the ER1 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPIN) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPIN must be positive. Correct the ER1 file and execute the program again.

Code: 2
Reserved

Code: 3
SZF? Local integration failed; IHLF=NN

Action: This error occurs during estimation of SZ over the source when no gas is present. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (SZSTP0 in the ER1 file). If this does not work, increase the error criteria for all of the dependent variables being integrated (SZERR in the ER1 file).

When IHLF=12, the initial increment of the independent variable (SZSTP0) is 0. Correct the ER1 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (SZSTP0) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. SZSTP0 must be positive. Correct the ER1 file and execute the program again.

Code: 4
SURFACE? Negative QRTE for positive DELTA_T.

Action: This is a diagnostic message indicating an error in estimation of the heat capacity. Check the input to the model and execute the program again.

Code: 5
CRFG? More points for GEN3 were needed.

Action: The COMMON area /GEN3/ stores representative values of the calculated source parameters. If this message occurs, relax the CRFG error criteria (CRFGER) in the ER1 file. If this is a common problem, the length of the /GEN3/ vectors can be increased by changing the value of MAXL in DEGADIS1.DEC and reinstalling DEGADIS.

Code: 6
TUPF? Observer calculations--TUPF failed

Action: The trial-and-error search associated with finding the upwind edge of the gas source for an observer failed. Often this problem can be avoided by adding one or two additional observers to the present number of observers (which changes the conditions for the trial and error). Another possibility is to increase the error criteria for this function (ERTUPF) in the ER2 file.

Code: 7
TDNF? Observer calculations--TDNF failed

Action: The trial-and-error search associated with finding the downwind edge of the gas source for an observer failed. Often this problem can be solved by adding one or two additional observers to the present number of observers (which changes the conditions for the trial and error). Another possibility is to increase the error criteria for this function (ERTDNF) in the ER2 file.

Code: 8
SSSUP? Observer Integration failed, IHLF=NN

Action: This error occurs during integration of the five differential equations which average the source for each observer. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (STPO in the ER2 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERRO in the ER2 file) or increase the error criteria for the variables violating the criteria by decreasing the error weight for that variable (one of the following: WTAIO, WTQOO, or WTSZO in the ER2 file).

When IHLF=12, the initial increment of the independent variable (STPO) is 0. Correct the ER2 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPO) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPO must be positive. Correct the ER2 file and execute the program again.

Code: 9
SSSUP/SDEGADIS2? Pseudosteady Integration failed, IHLF=NN

Action: This error occurs during integration of the four differential equations describing the portion of the downwind calculation when b > 0. The routine calling TRAP is SSSUP if a transient simulation is being executed; if a steady-state simulation is being executed, the calling routine is SDEGADIS2. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (STPP in the ER2 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERRP in the ER2 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (one of the following: WTSZP, WTSYP, WTBEP, or WTDH in the ER2 file).

When IHLF=12, the initial increment of the independent variable (STPP) is 0. Correct the ER2 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPP) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPP must be positive. Correct the ER2 file and execute the program again.

Code: 10
SSSUP/SDEGADIS2? Gaussian Integration fail, IHLF=nn

Action: This error occurs during integration of the differential equations describing the portion of the downwind calculation when b=0. The routine calling TRAP is SSSUP if a transient simulation is being executed; if a steady-state simulation is being executed, the calling routine is SDEGADIS2. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable. (STPG in the ER2 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERRG in the ER2 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (either WTRUH or WTDHG in the ER2 file).

When IHLF=12, the initial increment of the independent variable (STPG) is 0. Correct the ER2 file and execute the program again.

When IHLF=13, the initial increment of the dependent variable (STPG) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPG must be positive. Correct the ER2 file, and execute the program again.

Code: 11
SSSUP/SDEGADIS2 Total No. of Records exceeds 120,000

Action: This is an arbitrary stopping point for the process in order to keep a runaway simulation from filling up disk space. Relax the output specification (ODLP, ODLLP, ODLG, or, DLLG) in the ER2 file in order to generate less output if the input parameters are valid.

Code: 12
Reserved

Code: 13
Reserved

Code: 14
Reserved

Code: 15
Reserved

Code: 16
PSSOUT/PSSOUTSS? PSS started with B < 0.

Action: This condition is checked at the beginning of the downwind calculation in order to confirm proper handling of the movement to the Gaussian phase of the downwind calculation. Check the initial conditions and execute the program again.

Code: 17
TPROP/ADDHEAT? Enthalpy out of bounds

Action: Diagnostic message indicating that an enthalpy lower than the adiabatic mixing enthalpy was passed to ADDHEAT. Check the input conditions and execute the program again.

Code: 18

```
ALPH?  ALPHA integration failed, IHLF=NN
```

Action: The integration which determines the integral least squares fit for ALPHA has failed. Note that small values of the Monin-Obukhov length (ML < O(1m)) in combination with stable atmospheric conditions may cause this failure. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the absolute value of the initial step size of the independent variable (STPINZ in the ER1 file). If this does not work, it will be necessary to increase the error criteria (ERBNDZ in the ER1 file).

When IHLF=12, the initial increment of the independent variable (STPINZ) is 0. Correct the ER1 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPINZ) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPINZ must be negative. Correct the ER1 file and execute the program again. This error will also occur if the surface roughness ZR is greater than the reference height Z0.

Code: 19

```
ALPH?  RTMI has failed to locate ALPHA IERR:NN
```

Action: The search procedure which determines ALPHA has failed. This error may be the result of an unusual velocity specification such as small values of the Monin-Obukhov length (ML < O(1.m)) or small reference heights (Z0 < O(10. * ML)). IERR is an error code returned by the routine RTMI.

When IERR=1, the search for ALPHA failed after a specified number of iterations. Increase the error bound used by RTMI (EPS in the ER1 file).

When IERR=2, the basic assumption that the function which governs the search for ALPHA changes sign over the specified interval is false. Increase the search interval by decreasing the lower bound of ALPHA (XLI in the ER1 file) and increasing the upper bound (XRI in the ER1 file).

Code: 20

```
ESTRT?  Premature EOF in RUN_NAME.ER1 or RUN_NAME.ER2
```

Action: The portion of the program which reads ER1 and ER2 files encountered an end-of-file mark before all of the information had been read. Confirm these files and execute the program again. If necessary, copy and edit the appropriate EXAMPLE file and execute the program again.

Code: 21
ESTRT1/ESTRT2/ESTRT2SS/ESTRT3? DECODE failed.

Action: The portion of the program which reads the ER1, ER2, or the ER3 file failed to understand a numerical entry. The numbers must appear in columns 11-20 of the line with no alphabetic characters in the field. (Note that exponential notation is not allowed.) This restriction does not apply to comment lines which have an exclamation point (!) in the first column.

Code: 22
ESTRT1? The parameter file RUN_NAME.ER1 was not found.

Action: The ER1 file was not found for the current simulation (RUN_NAME). Copy the file EXAMPLE.ER1 to RUN_NAME.ER1 and edit it as necessary. Execute the program again.

Code: 23
SORTS1? Fewer than 3 points sorted for any time.

Action: Only one or two simulation points were applicable for the sort times specified. There are three possible causes for this condition:

(1) if this message appears when the sort times are defaulted (CHECK5 is set to 0. in the ER3 file), the number of observers will probably have to be increased to give a good resolution of the downwind concentration field. (The number of observers is NOBS in the ER2 file with a maximum (MAXNOB) given in DEGADIS2.DEC.). As a rule of thumb, one gets good resolution of the downwind concentration field if the ratio: (secondary source duration / number of observers) is less than about 10 seconds (or 20 at most).

(2) The sort times specified in the ER3 file were before the simulation had developed significantly. This is only applicable when the user is specifying the sort times (i.e. when CHECK5 is set to 1. in the ER3 file). Increase the time of the first sort (ERT1), and rerun the program.

(3) The sort times specified in the ER3 file were after the gas was below the lowest concentration of interest. This is only applicable when the user is specifying the sort times (i.e. when CHECK5 is set to 1. in the ER3 file). Increase the time of the first sort (ERT1), and rerun the program. If additional results are desired for later times, restart the simulation and specify a lower concentration of interest in the input step (lower CCLOW in DEGADISIN).

Code: 24
TPROP? Trial and error loop compromised.

Action: TPROP estimates the temperature of a mixture based upon the composition and enthalpy of the mixture. Ensure the properties for the diffusing species are entered correctly and execute the simulation again.

Code: 25
TPROP? Isothermal density loop compromised.

Action: This error should never occur, but if it does, rebuild the model from the original files and run the simulation over.

Code: 26
TPROP? Invalid entry flag in ADIABAT.

Action: This is a programming diagnostic and should never occur. If it does, rebuild the model from the original files.

Code: 27
Reserved

Code: 28
TPROP? IGEN request too large in SETDEN.

Action: The subroutine SETDEN (in TPROP) performs a series of adiabatic mixing calculations with a specified gas mixture and ambient air and places the result in the array DEN(5,IGEN). This error indicates more points are needed in DEN than were originally requested. Increase the allocation for DEN by changing the value of IGEN in DEGADISIN.DEC and reinstalling DEGADIS.

Code: 29
PHIF? Flag IPHIFL is out of bounds.

Action: Proper values of IPHIFL are integers between 1 and 5 inclusive. Although values of IPHIFL are entered in the ER1 file as real numbers, they should be in this range. Check the ER1 file and execute the program again.

Code: 30
SSSUP/SDEADIS2? Concentration greater than RHOE.

Action: If the concentration of the contaminant becomes greater than the pure component density for an isothermal simulation, this error will occur. However, this situation should never occur. Check the input conditions and execute the program again.

Code: 31
SSSUP? Concentration greater than RHOE.

Action: If the concentration of the contaminant becomes greater than the pure component density for an isothermal simulation, this error will occur. However, this situation should never occur. Check the input conditions and execute the program again.

Code: 32
PSS? Sz convergence failure.

Action: This is a programming diagnostic and should never occur. If it does, check the input conditions and execute the program again.

Code: 33
SSG? Sz convergence failure.

Action: This is a programming diagnostic and should never occur. If it does, check the input conditions and execute the program again.

APPENDIX J
OOMS/DEGADIS
EXAMPLE SIMULATION OUTPUT

(corresponds to example in Section IV of this volume)

```
O DOWNWIND DISTANCE TO FIRST PRINTOUT = 10.0 M
O DISTANCE BETWEEN PRINTOUTS = 10.0 M
O AMBIENT AIR DENSITY = 1.18 KG/M**3
O DISPERSION JET DENSITY = 2.33 KG/M**3
O AMBIENT TEMPERATURE = 298.0 DEG CELSIUS
O INITIAL JET TEMPERATURE = 298.0 DEG CELSIUS
O INITIAL JET DIAMETER =  20.0 CM
O INITIAL JET FLOWRATE =  0.2880E+01 M**3/S
O DEPTH OF SECTOR = 1000.0 M
O DISTANCE FROM THE CENTERLINE TO THE JET ORIGIN
  IN THE Y DIRECTION =     0.0 M
O DISTANCE FROM THE SECTOR SURFACE TO THE JET ORIGIN
  IN THE Z DIRECTION =  967.0 M
O MEAN VELOCITY OF THE WIND =  5.8 M/S
O ORIENTATION OF THE JET
     VERTICAL UPWARD
O ENTRAINMENT COEFFICIENTS
     ALFA 1=  0.0570
     ALFA 2=  0.5000
     ALFA 3=  1.0000


JET DEVELOPMENT:

    X              Z          CONCENTRATION(CL)
 (METERS)       (METERS)         (KG/M**3)

  10.7435        41.490         0.8797E-01
  20.5922        43.195         0.5794E-01
  30.5349        44.208         0.4645E-01
  40.5174        44.875         0.4044E-01
  50.5074        45.317         0.3688E-01
  60.5036        45.592         0.3470E-01
  70.5026        45.727         0.3343E-01
  80.5025        45.738         0.3275E-01
  90.5019        45.631         0.3173E-01
 100.4995        45.415         0.3030E-01
 110.4947        45.104         0.2859E-01
 120.4870        44.711         0.2673E-01
 130.4765        44.254         0.2485E-01
 140.4635        43.743         0.2302E-01
 150.4483        43.192         0.2128E-01
 160.4313        42.609         0.1967E-01
 170.4129        42.002         0.1820E-01
 180.3933        41.377         0.1685E-01
 190.3729        40.738         0.1564E-01
 200.3519        40.088         0.1453E-01
 210.3304        39.432         0.1354E-01
 220.3084        38.769         0.1264E-01
 230.2861        38.102         0.1182E-01
 240.2637        37.432         0.1108E-01
 250.2412        36.759         0.1040E-01
 260.2185        36.085         0.9788E-02
 270.1957        35.409         0.9226E-02
 280.1729        34.733         0.8712E-02
 290.1501        34.055         0.8239E-02
 300.1269        33.377         0.7804E-02
 310.1037        32.698         0.7403E-02
 320.0805        32.018         0.7032E-02
 330.0573        31.338         0.6688E-02
```

```
      340.0341         30.657        0.6369E-02
      350.0109         29.974        0.6072E-02
      360.9853         29.222        0.5768E-02
      370.9618         28.537        0.5511E-02
      380.9381         27.851        0.5271E-02
      390.9146         27.163        0.5045E-02
      400.8907         26.473        0.4834E-02
      410.8668         25.781        0.4635E-02
      420.8429         25.086        0.4447E-02
      430.8186         24.389        0.4270E-02
      440.7942         23.690        0.4103E-02
      450.7694         22.987        0.3945E-02
      460.7444         22.281        0.3795E-02
      470.7195         21.571        0.3653E-02
      480.6941         20.857        0.3518E-02
      490.6682         20.138        0.3390E-02
      500.6421         19.415        0.3267E-02
      510.6154         18.686        0.3151E-02
      520.5886         17.952        0.3040E-02
      530.5610         17.211        0.2933E-02
      540.5332         16.464        0.2832E-02

********************      U O A _ D E G A D I S   M O D E L   O U T P U T   - -   V E R S I O N   2.0      ********************

***************               *************** 17-MAR-1988 10:42:30.90 ***************               ***************

Data input on                  17-MAR-1988 10:42:27.45
Source program run on          17-MAR-1988 10:42:30.90


          ******************************************************************************************************
          *                                                                                                    *
          *       NOTE:                                                                                        *
          *       -----                                                                                        *
          *                                                                                                    *
          *       >    All Calculations are limited to circular liquid sources.                                *
          *                                                                                                    *
          ******************************************************************************************************


          TITLE BLOCK

Steady-state simulation test  OOMS-DEGADIS  MIC release (Bhopal)


     Wind velocity at reference height                          2.90  m/s
     Reference height                                          10.00  m

     Surface roughness length                              0.100      m

     Pasquill Stability class                                F

     Monin-Obukhov length                                   17.5      m
     Gaussian distribution constants     Delta              0.03600   m
                                         Beta               0.89400

     Wind velocity power law constant    Alpha              0.44905
     Friction velocity                                      0.13910   m/s
```

```
Ambient Temperature                                    298.00  K

Surface Temperature                                    298.00  K
Ambient Pressure                                        1.000  atm
Ambient Absolute Humidity                           1.009E-02  kg/kg BDA
Ambient Relative Humidity                               50.00  %

Adiabatic Mixing:   Mole fraction   CONCENTRATION OF C    GAS DENSITY       Enthalpy         Temperature
                                        kg/m**3             kg/m**3           J/kg              K
                       0.00000          0.00000             1.17737       0.00000E+00        298.00
                       0.06729          0.15688             1.25501       0.00000E+00        298.00
                       0.14408          0.33590             1.34362       0.00000E+00        298.00

                       0.23254          0.54213             1.44569       0.00000E+00        298.00
                       0.33555          0.78227             1.56454       0.00000E+00        298.00
                       0.45702          1.06543             1.70469       0.00000E+00        298.00

                       0.60239          1.40431             1.87241       0.00000E+00        298.00
                       0.77950          1.81714             2.07674       0.00000E+00        298.00
                       1.00000          2.33112             2.33112       0.00000E+00        298.00

Specified Gas Properties:

   Molecular weight:                                          57.000
   Storage temperature:                                       298.00     K
   Density at storage temperature and ambient pressure:      2.3311     kg/m**3
   Mean heat capacity constant:                               80700.
   Mean heat capacity power:                                  1.0000
   Upper mole fraction contour:                              3.00000E-05
   Lower mole fraction contour:                              2.00000E-05
   Height for isopleths:                                     0.50000     m

Source input data points

          Initial mass in cloud:   0.00000E+00

   Time        Contaminant      Source Radius     Contaminant       Temperature       Enthalpy
                Mass Rate                        Mass Fraction
     s            kg/s                m        kg contam/kg mix         K               J/kg
0.00000E+00      6.7200            15.893        2.34446E-03         298.00        0.00000E+00
  6023.0         6.7200            15.893        2.34446E-03         298.00        0.00000E+00
  6024.0       0.00000E+00      0.00000E+00      2.34446E-03         298.00        0.00000E+00
  6025.0       0.00000E+00      0.00000E+00      2.34446E-03         298.00        0.00000E+00

   Calculation procedure for ALPHA:  1

   Entrainment prescription for PHI:  3

   Layer thickness ratio used for average depth:    2.1500

   Air entrainment coefficient used: 0.590

   Gravity slumping velocity coefficient used: 1.150

   NON Isothermal calculation

   Heat transfer calculated with correlation:  1

   Water transfer not included
```

```
          *****                    CALCULATED SOURCE PARAMETERS                    *****

    Time        Gas Radius      Height        Qstar       SZ(x=L/2.)   Mole frac C     Density    Temperature     Rich No.
    sec             m              m        kg/m**2/s         m                        kg/m**3         K

 0.000000E+00    15.8926     1.100000E-05  1.491725E-04   1.83550     1.185348E-03   1.17873      298.000       0.756144
  7.68000        15.8926       23.1214     1.491725E-04   29.8055     1.185348E-03   1.17873      298.000       0.756144

 Source strength [kg/s] :                    6.7200    Equivalent Primary source radius [m] :         15.893
 Equivalent Primary source length [m] :      31.785    Equivalent Primary source half-width [m] :     12.482

 Secondary source concentration [kg/m**3] :  2.76350E-03  Secondary source SZ [m] :                   29.806

 Contaminant flux rate:   6.65150E-03


 Secondary source mass fractions... contaminant:  2.344463E-03   air:   0.98769
           Enthalpy:   0.00000E+00      Density:    1.1787

 Secondary source length [m] :               31.785    Secondary source half-width [m] :              12.482


  Distance     Mole    Concentration Density       Gamma     Temperature   Half       Sz         Sy       Width at z=  0.50 m to:
              Fraction                                                    Width                          2.000E-03mole% 3.000E-03mole%
    (m)                 (kg/m**3)  (kg/m**3)                    (K)        (m)       (m)        (m)         (m)          (m)

   15.9     1.185E-03  2.763E-03    1.1787      0.495        298.       12.5       29.8     3.000E-02    12.5         12.5
   16.7     1.185E-03  2.763E-03    1.1787      0.495        298.       11.9       29.8     0.727        13.3         13.3
   19.1     1.185E-03  2.762E-03    1.1787      0.495        298.       11.2       29.8      1.45        14.2         14.0

   22.3     1.184E-03  2.760E-03    1.1787      0.495        298.       10.7       29.8      2.05        14.8         14.6
   28.7     1.182E-03  2.756E-03    1.1787      0.495        298.       9.93       29.8      2.91        15.8         15.5
   35.1     1.181E-03  2.754E-03    1.1787      0.495        298.       9.35       29.9      3.56        16.5         16.2

   41.5     1.179E-03  2.749E-03    1.1787      0.495        298.       8.87       29.9      4.11        17.2         16.7
   67.1     1.173E-03  2.735E-03    1.1787      0.495        298.       7.36       30.1      5.81        19.1         18.5
   79.9     1.169E-03  2.725E-03    1.1787      0.495        298.       6.75       30.1      6.50        19.9         19.2

   92.7     1.165E-03  2.716E-03    1.1787      0.495        298.       6.20       30.2      7.12        20.5         19.8
   105.     1.161E-03  2.706E-03    1.1787      0.495        298.       5.70       30.3      7.69        21.2         20.4
   118.     1.157E-03  2.697E-03    1.1787      0.495        298.       5.23       30.3      8.22        21.8         20.9

   131.     1.153E-03  2.688E-03    1.1787      0.495        298.       4.78       30.4      8.72        22.3         21.4
   144.     1.149E-03  2.678E-03    1.1787      0.495        298.       4.37       30.5      9.19        22.9         21.9
   157.     1.145E-03  2.669E-03    1.1787      0.495        298.       3.97       30.6      9.64        23.3         22.4

   169.     1.141E-03  2.660E-03    1.1787      0.495        298.       3.59       30.6      10.1        23.8         22.8
   182.     1.137E-03  2.651E-03    1.1787      0.495        298.       3.22       30.7      10.5        24.3         23.2
   195.     1.133E-03  2.642E-03    1.1787      0.495        298.       2.87       30.8      10.9        24.7         23.6

   208.     1.129E-03  2.633E-03    1.1787      0.495        298.       2.54       30.8      11.3        25.1         24.0
   221.     1.126E-03  2.624E-03    1.1787      0.495        298.       2.21       30.9      11.6        25.5         24.3
   233.     1.122E-03  2.615E-03    1.1787      0.495        298.       1.89       31.0      12.0        25.9         24.7

   246.     1.118E-03  2.607E-03    1.1787      0.495        298.       1.58       31.1      12.3        26.3         25.0
   259.     1.114E-03  2.598E-03    1.1787      0.495        298.       1.29       31.1      12.7        26.7         25.4
   272.     1.111E-03  2.589E-03    1.1786      0.495        298.      0.993       31.2      13.0        27.0         25.7

   285.     1.107E-03  2.581E-03    1.1786      0.495        298.      0.709       31.3      13.3        27.4         26.0
   297.     1.103E-03  2.572E-03    1.1786      0.495        298.      0.431       31.3      13.6        27.7         26.3
   310.     1.100E-03  2.564E-03    1.1786      0.495        298.      0.160       31.4      13.9        28.0         26.6
   323.     1.096E-03  2.555E-03   1.1786   0.495    298.    0.000E+00  31.5   14.1   28.1   26.7
   342.     1.059E-03  2.468E-03   1.1786   0.495    298.    0.000E+00  31.6   14.5   28.9   27.4
   361.     1.024E-03  2.386E-03   1.1785   0.495    298.    0.000E+00  31.7   15.0   29.7   28.1

   387.     9.779E-04  2.280E-03   1.1785   0.495    298.    0.000E+00  31.9   15.6   30.7   29.0
   400.     9.564E-04  2.230E-03   1.1785   0.495    298.    0.000E+00  31.9   15.8   31.2   29.5
   425.     9.157E-04  2.135E-03   1.1784   0.495    298.    0.000E+00  32.1   16.4   32.1   30.4

   451.     8.779E-04  2.047E-03   1.1784   0.495    298.    0.000E+00  32.3   17.0   33.1   31.3
   477.     8.426E-04  1.964E-03   1.1783   0.495    298.    0.000E+00  32.4   17.6   34.0   32.1
   502.     8.097E-04  1.888E-03   1.1783   0.495    298.    0.000E+00  32.6   18.2   35.0   33.0

   528.     7.788E-04  1.816E-03   1.1783   0.495    298.    0.000E+00  32.8   18.8   35.9   33.9
   553.     7.499E-04  1.748E-03   1.1782   0.495    298.    0.000E+00  32.9   19.3   36.8   34.7
   579.     7.226E-04  1.685E-03   1.1782   0.495    298.    0.000E+00  33.1   19.9   37.7   35.5

   605.     6.970E-04  1.625E-03   1.1782   0.495    298.    0.000E+00  33.3   20.5   38.6   36.3
   630.     6.728E-04  1.569E-03   1.1781   0.495    298.    0.000E+00  33.5   21.1   39.5   37.1
   656.     6.499E-04  1.515E-03   1.1781   0.495    298.    0.000E+00  33.6   21.6   40.4   37.9

   681.     6.283E-04  1.465E-03   1.1781   0.495    298.    0.000E+00  33.8   22.2   41.2   38.7
   707.     6.077E-04  1.417E-03   1.1781   0.495    298.    0.000E+00  34.0   22.8   42.1   39.5
   733.     5.882E-04  1.371E-03   1.1780   0.495    298.    0.000E+00  34.2   23.3   42.9   40.2

   758.     5.697E-04  1.328E-03   1.1780   0.495    298.    0.000E+00  34.4   23.9   43.7   41.0
   784.     5.521E-04  1.287E-03   1.1780   0.495    298.    0.000E+00  34.6   24.5   44.5   41.7
   809.     5.353E-04  1.248E-03   1.1780   0.495    298.    0.000E+00  34.8   25.0   45.3   42.5

   835.     5.193E-04  1.211E-03   1.1780   0.495    298.    0.000E+00  35.0   25.6   46.1   43.2
   861.     5.040E-04  1.175E-03   1.1779   0.495    298.    0.000E+00  35.2   26.1   46.9   43.9
   899.     4.823E-04  1.124E-03   1.1779   0.495    298.    0.000E+00  35.5   27.0   48.1   44.9

   937.     4.621E-04  1.077E-03   1.1779   0.495    298.    0.000E+00  35.8   27.8   49.2   45.9
   976.     4.431E-04  1.033E-03   1.1779   0.495    298.    0.000E+00  36.1   28.6   50.4   46.9
 1.014E+03  4.252E-04  9.913E-04   1.1779   0.494    298.    0.000E+00  36.5   29.4   51.5   47.9

 1.053E+03  4.084E-04  9.522E-04   1.1778   0.494    298.    0.000E+00  36.8   30.3   52.5   48.9
 1.091E+03  3.926E-04  9.153E-04   1.1778   0.494    298.    0.000E+00  37.1   31.1   53.6   49.8
 1.129E+03  3.777E-04  8.805E-04   1.1778   0.494    298.    0.000E+00  37.4   31.9   54.7   50.7

 1.168E+03  3.636E-04  8.477E-04   1.1778   0.494    298.    0.000E+00  37.8   32.7   55.7   51.6
 1.206E+03  3.503E-04  8.167E-04   1.1778   0.494    298.    0.000E+00  38.1   33.5   56.7   52.5
 1.245E+03  3.377E-04  7.873E-04   1.1778   0.494    298.    0.000E+00  38.4   34.3   57.7   53.4

 1.283E+03  3.257E-04  7.594E-04   1.1777   0.494    298.    0.000E+00  38.8   35.1   58.7   54.2
 1.321E+03  3.144E-04  7.330E-04   1.1777   0.494    298.    0.000E+00  39.1   35.9   59.6   55.1
 1.360E+03  3.036E-04  7.079E-04   1.1777   0.494    298.    0.000E+00  39.5   36.7   60.6   55.9

 1.398E+03  2.934E-04  6.840E-04   1.1777   0.494    298.    0.000E+00  39.8   37.5   61.5   56.7
 1.437E+03  2.837E-04  6.613E-04   1.1777   0.494    298.    0.000E+00  40.2   38.3   62.4   57.4
 1.475E+03  2.744E-04  6.397E-04   1.1777   0.494    298.    0.000E+00  40.5   39.1   63.3   58.2

 1.526E+03  2.627E-04  6.124E-04   1.1777   0.494    298.    0.000E+00  41.0   40.2   64.5   59.2
 1.577E+03  2.517E-04  5.869E-04   1.1777   0.494    298.    0.000E+00  41.5   41.2   65.6   60.1
 1.629E+03  2.414E-04  5.628E-04   1.1776   0.494    298.    0.000E+00  42.0   42.3   66.7   61.1

 1.680E+03  2.317E-04  5.401E-04   1.1776   0.494    298.    0.000E+00  42.4   43.3   67.8   62.0
 1.731E+03  2.225E-04  5.188E-04   1.1776   0.494    298.    0.000E+00  42.9   44.4   68.9   62.8
 1.782E+03  2.139E-04  4.986E-04   1.1776   0.494    298.    0.000E+00  43.4   45.4   69.9   63.7

 1.833E+03  2.057E-04  4.796E-04   1.1776   0.494    298.    0.000E+00  43.9   46.5   70.9   64.5
 1.885E+03  1.980E-04  4.616E-04   1.1776   0.494    298.    0.000E+00  44.4   47.5   71.9   65.2
 1.936E+03  1.907E-04  4.446E-04   1.1776   0.494    298.    0.000E+00  44.9   48.5   72.9   66.0
```

```
1.987E+03  1.838E-04  4.284E-04   1.1776   0.494    298.   0.000E+00  45.4   49.6   73.8   66.7
2.038E+03  1.772E-04  4.131E-04   1.1776   0.494    298.   0.000E+00  45.9   50.6   74.7   67.4
2.089E+03  1.710E-04  3.986E-04   1.1776   0.494    298.   0.000E+00  46.4   51.6   75.6   68.1

2.153E+03  1.636E-04  3.815E-04   1.1776   0.494    298.   0.000E+00  47.0   52.9   76.7   68.9
2.217E+03  1.567E-04  3.654E-04   1.1775   0.494    298.   0.000E+00  47.6   54.2   77.7   69.7
2.281E+03  1.502E-04  3.502E-04   1.1775   0.494    298.   0.000E+00  48.3   55.5   78.7   70.4

2.345E+03  1.441E-04  3.360E-04   1.1775   0.494    298.   0.000E+00  48.9   56.7   79.7   71.0
2.409E+03  1.384E-04  3.226E-04   1.1775   0.494    298.   0.000E+00  49.5   58.0   80.6   71.7
2.473E+03  1.329E-04  3.100E-04   1.1775   0.494    298.   0.000E+00  50.2   59.3   81.5   72.3

2.537E+03  1.278E-04  2.980E-04   1.1775   0.493    298.   0.000E+00  50.8   60.5   82.4   72.8
2.601E+03  1.230E-04  2.867E-04   1.1775   0.493    298.   0.000E+00  51.4   61.8   83.2   73.4
2.665E+03  1.184E-04  2.760E-04   1.1775   0.493    298.   0.000E+00  52.1   63.0   84.0   73.8

2.729E+03  1.141E-04  2.659E-04   1.1775   0.493    298.   0.000E+00  52.7   64.3   84.8   74.3
2.806E+03  1.092E-04  2.545E-04   1.1775   0.493    298.   0.000E+00  53.5   65.8   85.7   74.7
2.883E+03  1.046E-04  2.438E-04   1.1775   0.493    298.   0.000E+00  54.2   67.3   86.5   75.1

2.960E+03  1.002E-04  2.337E-04   1.1775   0.493    298.   0.000E+00  55.0   68.8   87.3   75.5
3.037E+03  9.617E-05  2.242E-04   1.1775   0.493    298.   0.000E+00  55.8   70.3   88.0   75.8
3.113E+03  9.233E-05  2.153E-04   1.1775   0.493    298.   0.000E+00  56.5   71.7   88.7   76.0

3.190E+03  8.872E-05  2.068E-04   1.1775   0.493    298.   0.000E+00  57.3   73.2   89.3   76.2
3.267E+03  8.531E-05  1.989E-04   1.1775   0.493    298.   0.000E+00  58.1   74.7   89.9   76.3
3.344E+03  8.208E-05  1.914E-04   1.1775   0.493    298.   0.000E+00  58.8   76.2   90.5   76.4

3.421E+03  7.903E-05  1.843E-04   1.1775   0.493    298.   0.000E+00  59.6   77.6   91.0   76.4
3.497E+03  7.615E-05  1.775E-04   1.1775   0.492    298.   0.000E+00  60.4   79.1   91.4   76.3
3.574E+03  7.341E-05  1.712E-04   1.1775   0.492    298.   0.000E+00  61.1   80.5   91.8   76.2

3.651E+03  7.082E-05  1.651E-04   1.1774   0.492    298.   0.000E+00  61.9   82.0   92.2   76.0
3.728E+03  6.836E-05  1.594E-04   1.1774   0.492    298.   0.000E+00  62.7   83.5   92.5   75.7
3.805E+03  6.602E-05  1.539E-04   1.1774   0.492    298.   0.000E+00  63.4   84.9   92.8   75.4

3.881E+03  6.380E-05  1.487E-04   1.1774   0.492    298.   0.000E+00  64.2   86.4   93.0   75.0
3.958E+03  6.168E-05  1.438E-04   1.1774   0.492    298.   0.000E+00  64.9   87.8   93.2   74.5
4.035E+03  5.967E-05  1.391E-04   1.1774   0.492    298.   0.000E+00  65.7   89.3   93.3   74.0

4.112E+03  5.775E-05  1.346E-04   1.1774   0.492    298.   0.000E+00  66.5   90.7   93.4   73.3
4.189E+03  5.592E-05  1.304E-04   1.1774   0.492    298.   0.000E+00  67.2   92.1   93.4   72.7
4.265E+03  5.418E-05  1.263E-04   1.1774   0.491    298.   0.000E+00  68.0   93.6   93.4   71.9

4.342E+03  5.252E-05  1.224E-04   1.1774   0.491    298.   0.000E+00  68.7   95.0   93.3   71.0
4.419E+03  5.092E-05  1.187E-04   1.1774   0.491    298.   0.000E+00  69.5   96.4   93.2   70.1
4.496E+03  4.940E-05  1.152E-04   1.1774   0.491    298.   0.000E+00  70.2   97.9   93.0   69.1

4.573E+03  4.795E-05  1.118E-04   1.1774   0.491    298.   0.000E+00  71.0   99.3   92.8   67.9
4.649E+03  4.656E-05  1.085E-04   1.1774   0.491    298.   0.000E+00  71.7   101.   92.5   66.7
4.726E+03  4.522E-05  1.054E-04   1.1774   0.491    298.   0.000E+00  72.5   102.   92.2   65.4

4.803E+03  4.394E-05  1.024E-04   1.1774   0.491    298.   0.000E+00  73.2   104.   91.8   63.9
4.880E+03  4.272E-05  9.959E-05   1.1774   0.491    298.   0.000E+00  74.0   105.   91.4   62.3
4.957E+03  4.154E-05  9.685E-05   1.1774   0.490    298.   0.000E+00  74.7   106.   90.9   60.6

5.033E+03  4.041E-05  9.422E-05   1.1774   0.490    298.   0.000E+00  75.5   108.   90.4   58.8
5.110E+03  3.933E-05  9.170E-05   1.1774   0.490    298.   0.000E+00  76.2   109.   89.7   56.7
5.187E+03  3.829E-05  8.927E-05   1.1774   0.490    298.   0.000E+00  76.9   111.   89.1   54.6

5.264E+03  3.729E-05  8.694E-05   1.1774   0.490    298.   0.000E+00  77.7   112.   88.4   52.2
5.341E+03  3.633E-05  8.469E-05   1.1774   0.490    298.   0.000E+00  78.4   113.   87.6   49.5
```

```
5.417E+03  3.540E-05  8.253E-05    1.1774    0.490      298.     0.000E+00  79.1      115.      86.7      46.6

5.494E+03  3.451E-05  8.046E-05    1.1774    0.489      298.     0.000E+00  79.9      116.      85.8      43.4
5.571E+03  3.365E-05  7.845E-05    1.1774    0.489      298.     0.000E+00  80.6      118.      84.8      39.7
5.648E+03  3.282E-05  7.653E-05    1.1774    0.489      298.     0.000E+00  81.3      119.      83.7      35.6

5.725E+03  3.203E-05  7.467E-05    1.1774    0.489      298.     0.000E+00  82.1      120.      82.6      30.6
5.801E+03  3.126E-05  7.287E-05    1.1774    0.489      298.     0.000E+00  82.8      122.      81.3      24.5
5.878E+03  3.052E-05  7.115E-05    1.1774    0.489      298.     0.000E+00  83.5      123.      80.0      15.8

5.955E+03  2.980E-05  6.948E-05    1.1774    0.489      298.     0.000E+00  84.3      125.      78.6
6.032E+03  2.911E-05  6.786E-05    1.1774    0.488      298.     0.000E+00  85.0      126.      77.1
6.109E+03  2.844E-05  6.631E-05    1.1774    0.488      298.     0.000E+00  85.7      127.      75.5

6.185E+03  2.780E-05  6.480E-05    1.1774    0.488      298.     0.000E+00  86.4      129.      73.8
6.262E+03  2.717E-05  6.335E-05    1.1774    0.488      298.     0.000E+00  87.1      130.      71.9
6.339E+03  2.657E-05  6.195E-05    1.1774    0.488      298.     0.000E+00  87.9      131.      70.0

6.416E+03  2.599E-05  6.059E-05    1.1774    0.488      298.     0.000E+00  88.6      133.      67.9
6.493E+03  2.542E-05  5.927E-05    1.1774    0.488      298.     0.000E+00  89.3      134.      65.7
6.569E+03  2.488E-05  5.800E-05    1.1774    0.487      298.     0.000E+00  90.0      136.      63.3

6.646E+03  2.435E-05  5.676E-05    1.1774    0.487      298.     0.000E+00  90.7      137.      60.7
6.723E+03  2.384E-05  5.557E-05    1.1774    0.487      298.     0.000E+00  91.4      138.      57.9
6.800E+03  2.334E-05  5.441E-05    1.1774    0.487      298.     0.000E+00  92.1      140.      54.8

6.877E+03  2.286E-05  5.329E-05    1.1774    0.487      298.     0.000E+00  92.8      141.      51.5
6.953E+03  2.239E-05  5.220E-05    1.1774    0.486      298.     0.000E+00  93.5      142.      47.8
7.030E+03  2.194E-05  5.115E-05    1.1774    0.486      298.     0.000E+00  94.2      144.      43.6

7.107E+03  2.150E-05  5.013E-05    1.1774    0.486      298.     0.000E+00  94.9      145.      38.9
7.184E+03  2.107E-05  4.913E-05    1.1774    0.486      298.     0.000E+00  95.6      147.      33.4
7.261E+03  2.066E-05  4.817E-05    1.1774    0.486      298.     0.000E+00  96.3      148.      26.5

7.337E+03  2.026E-05  4.723E-05    1.1774    0.486      298.     0.000E+00  97.0      149.      16.6
7.414E+03  1.987E-05  4.633E-05    1.1774    0.485      298.     0.000E+00  97.7      151.
7.491E+03  1.949E-05  4.544E-05    1.1774    0.485      298.     0.000E+00  98.4      152.

7.568E+03  1.912E-05  4.458E-05    1.1774    0.485      298.     0.000E+00  99.1      153.
7.645E+03  1.877E-05  4.375E-05    1.1774    0.485      298.     0.000E+00  99.8      155.
7.721E+03  1.842E-05  4.294E-05    1.1774    0.485      298.     0.000E+00  101.      156.

7.798E+03  1.808E-05  4.215E-05    1.1774    0.484      298.     0.000E+00  101.      157.
7.875E+03  1.775E-05  4.139E-05    1.1774    0.484      298.     0.000E+00  102.      159.
7.952E+03  1.743E-05  4.064E-05    1.1774    0.484      298.     0.000E+00  103.      160.

8.029E+03  1.712E-05  3.991E-05    1.1774    0.484      298.     0.000E+00  103.      161.
8.105E+03  1.682E-05  3.921E-05    1.1774    0.484      298.     0.000E+00  104.      163.
8.182E+03  1.652E-05  3.852E-05    1.1774    0.484      298.     0.000E+00  105.      164.

8.259E+03  1.623E-05  3.785E-05    1.1774    0.483      298.     0.000E+00  105.      165.
8.336E+03  1.595E-05  3.719E-05    1.1774    0.483      298.     0.000E+00  106.      167.
8.413E+03  1.568E-05  3.656E-05    1.1774    0.483      298.     0.000E+00  107.      168.

8.489E+03  1.541E-05  3.594E-05    1.1774    0.483      298.     0.000E+00  107.      169.
8.566E+03  1.516E-05  3.533E-05    1.1774    0.482      298.     0.000E+00  108.      171.
8.643E+03  1.490E-05  3.474E-05    1.1774    0.482      298.     0.000E+00  109.      172.

8.720E+03  1.466E-05  3.417E-05    1.1774    0.482      298.     0.000E+00  109.      173.
8.797E+03  1.442E-05  3.361E-05    1.1774    0.482      298.     0.000E+00  110.      175.
8.873E+03  1.418E-05  3.306E-05    1.1774    0.482      298.     0.000E+00  111.      176.

8.950E+03  1.395E-05  3.253E-05    1.1774    0.481      298.     0.000E+00  111.      177.
```

```
9.027E+03  1.373E-05  3.201E-05    1.1774     0.481      298.     0.000E+00   112.      179.
9.104E+03  1.351E-05  3.150E-05    1.1774     0.481      298.     0.000E+00   113.      180.

9.181E+03  1.330E-05  3.100E-05    1.1774     0.481      298.     0.000E+00   113.      181.
9.257E+03  1.309E-05  3.052E-05    1.1774     0.481      298.     0.000E+00   114.      183.
9.334E+03  1.289E-05  3.005E-05    1.1774     0.480      298.     0.000E+00   115.      184.

9.411E+03  1.269E-05  2.958E-05    1.1774     0.480      298.     0.000E+00   115.      185.
9.488E+03  1.250E-05  2.913E-05    1.1774     0.480      298.     0.000E+00   116.      187.
9.565E+03  1.231E-05  2.869E-05    1.1774     0.480      298.     0.000E+00   117.      188.

9.641E+03  1.212E-05  2.826E-05    1.1774     0.479      298.     0.000E+00   117.      189.
9.718E+03  1.194E-05  2.784E-05    1.1774     0.479      298.     0.000E+00   118.      191.
9.795E+03  1.176E-05  2.742E-05    1.1774     0.479      298.     0.000E+00   119.      192.

9.872E+03  1.159E-05  2.702E-05    1.1774     0.479      298.     0.000E+00   119.      193.
9.949E+03  1.142E-05  2.663E-05    1.1774     0.478      298.     0.000E+00   120.      195.
1.003E+04  1.126E-05  2.624E-05    1.1774     0.478      298.     0.000E+00   121.      196.

1.010E+04  1.109E-05  2.586E-05    1.1774     0.478      298.     0.000E+00   121.      197.
1.018E+04  1.094E-05  2.549E-05    1.1774     0.478      298.     0.000E+00   122.      199.
1.026E+04  1.078E-05  2.513E-05    1.1774     0.477      298.     0.000E+00   123.      200.

1.033E+04  1.063E-05  2.478E-05    1.1774     0.477      298.     0.000E+00   123.      201.
1.041E+04  1.048E-05  2.443E-05    1.1774     0.477      298.     0.000E+00   124.      202.
1.049E+04  1.033E-05  2.409E-05    1.1774     0.477      298.     0.000E+00   125.      204.

1.056E+04  1.019E-05  2.376E-05    1.1774     0.476      298.     0.000E+00   125.      205.
1.064E+04  1.005E-05  2.344E-05    1.1774     0.476      298.     0.000E+00   126.      206.
1.068E+04  9.984E-06  2.328E-05    1.1774     0.476      298.     0.000E+00   126.      207.


For the UFL of   3.00000E-03 mole percent, and the LFL of   2.00000E-03 mole percent:

The mass of contaminant between the UFL and LFL is:   1213.3     kg.
The mass of contaminant above the LFL is:    6662.1     kg.
```